COMPARATIVE STUDY OF INFLUENCE OF IMAGE SIZE ON THE PERFORMANCE TESTING USING LOADRUNNER

OMRAN MAKI ABDELSALAM

A dissertation report submitted in partial
fulfilment of the requirement for the award of the
Degree of Master of Computer Science (Software Engineering)

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

December 2015

ABSTRACT

Software testing is an important stage in software life cycle and it is an assurance of software quality. Software testing exists in each stage of software life cycle and verifies whether the expected results are achieved or not and corrects the bugs as soon as possible. In software development processing, bugs always exist no matter what technology is adopted. Testing is applied to find bugs and used to calculate software bugs density. Performance testing is one of the vital activities spanning the whole life cycle of software engineering. This is also to improve the quality and reliability of web applications. In recent years World Wide Web traffic has shown phenomenal growth. The main causes are the continuing increase in the number of people navigating the Internet and the creation of millions of new Web sites. In addition, the structure of Web pages has become more complex, including not only HTML files but also other components. This has affected both the download times of Web pages and the network bandwidth required. The aim of this dissertation report is to monitor performance parameters which are the successful request rates and response time of web application, by using Loadrunner testing tool in three scenarios with different sizes of images and find out how images influence the access time of the web pages. The experimental results of the Loadrunner in the three scenarios of online shopping system showed that the size of images and the number of users affected the web application performance. The results proved that the increasing size of images and the number of users led to the falling of the successful request rate system and the increasing of the average response time**.**

ABSTRAK

Pengujian perisian adalah satu peringkat yang penting dalam kitaran hidup perisian dan ia adalah satu jaminan ke atas kualiti perisian. Pengujian perisian wujud dalam setiap peringkat kitaran hayat perisian dan mengesahkan sama ada keputusan yang dijangkakan dapat dicapai atau tidak, serta menyelenggara pepijat seberapa segera yang mungkin. Dalam proses membangunkan perisian, pepijat sentiasa wujud tidak kira apa teknologi yang diguna pakai. Pengujian digunakan untuk mencari pepijat dan digunakan untuk mengira kepadatan pepijat perisian. Ujian prestasi adalah salah satu aktiviti penting yang merangkumi kitar hayat keseluruhan kejuruteraan perisian. Ia juga bagi meningkatkan kualiti dan kebolehpercayaan aplikasi web. Sejak kebelakangan ini, penggunaan internet telah menunjukkan pertumbuhan yang luar biasa. Punca utama adalah peningkatan yang berterusan dalam bilangan pengguna yang melayari Internet dan rekaan berbagai-bagai laman web baru. Di samping itu, struktur laman web telah menjadi lebih kompleks, termasuk fail-fail HTML dan juga komponen-komponen lain. Ini telah memberi kesan kepada jumlah muat turun dari laman web dan juga rangkaian jalur lebar yang diperlukan. Tujuan laporan disertasi ini adalah untuk memantau parameter prestasi iaitu kadar kejayaan sesuatu tindakan dan masa yang diperlukan untuk aplikasi web dengan menggunakan alat ujian Loadrunner dalam tiga senario yang melibatkan saiz imej yang berbeza. Laporan ini juga bertujuan untuk mengetahui bagaimana imej mempengaruhi tempoh masa akses di laman-laman web. Keputusan eksperimen penggunaan Loadrunner untuk tiga senario sistem membeli-belah dalam talian menunjukkan bahawa saiz imej dan bilangan pengguna memberi kesan kepada prestasi aplikasi web. Keputusan membuktikan bahawa peningkatan saiz imej dan jumlah pengguna menyumbang kepada penurunan kadar kejayaan sesuatu tindakan dan meningkatkan purata masa tindak balas.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

DB	–	Database
Vuser	–	Virtual User
VuGen	–	Virtual User Generator
HTTP	–	Hypertext Transfer Protocol
HTML	–	Hypertext Markup Language
RT	–	Response Time
SR	–	Success Requests
HP	–	Hewlett-Packard

CHAPTER 1

INTRODUCTION

This chapter provides the overview of the research done. First of all it describes the background of the research problem. Next, it elaborates the research problems that shape the direction of this research. Then, it presents the objectives and scope of the research. Finally, outlines of the dissertation report.

1.1 Background of Study

The massive growth and development of web applications and web-based services have changed the topography and the ways in information flows. This has been a substantial effect whether to the government, corporate, private services, educational and research centres. This has resulted web applications that have been developed in a user friendly interface. From here, people can see how important is a developer's work and the creativity could be unlimited. Other than being user friendly and having a good feature, the web application must also be stable and able to handle a large number of users at the same time. To ensure confidence level on the web application reliability during peak time, the web application must be tested. Testing stage is one of the most crucial stages in software and web application life cycle as the reliability affects the final product performance once it has been released in the market (Kalita & Bezboruah, 2011).

Testing stage is one of the most important stages in the software development life cycle. Software testing is a repeating loop in each stage of the development life cycle. The web application software will be tested continuously to fix the bugs, the performance of the software and the stability as well. The web application will always be updated from time to time as the developers have fixed some of the bugs in the web application. It is important to know that bugs always exist in technology and it is important to detect the bugs with calculation of bugs' density (Chakrapani & Ramesh, 2011). From the various software testing parameters available, this dissertation report focuses on performance testing on web applications.

Web application usually needs to concurrently serve many users while the developer could be working in a small group or even work alone. This creates an issue where the report and request of the users on bugs are very difficult to be fulfilled in a short time. Thus, an automatic testing system is required to manage the traffic (Yunming & Xu, 2009). In addition, as the web application's functions are getting more complicated and more complex to cater the technological growth, many application performance analysis methods are obsolete. This is due to the technological environment becoming more dynamic and unpredictable for the users. Load testing is a great choice in web application performance analysis to simulate the user access and the web application performance as it is low cost, high flexibility and better simulation ability. In the process of using conventional load testing to analyse web application performance, it is a must to separate load tests for hardware configuration due to numerous hardware configurations in the world (Lu, Wu & Wang, 2006).

Load testing is testing the system performance in one level of a load. The load level can be defined as the number of web users at a time and the amount of online processing data at a time (Yunming & Xu, 2009). For example, it testifies how many users the web can support at the same time and the scenarios where extra numbers of users are added to the traffic. Web page also affects web load, since it consists of many file types such as hypertext markup language (HTML), text, JavaScript, video and image. Although there are many causes affecting the web load, this dissertation report has focused on the increase of the image size and the number of concurrent users by using performance testing tool which is the Loadrunner.

1.2 Problem Statement

In the case of web applications, system performance is a critical issue because web users are not able to wait too long for a response to their requests. Besides, they also expect that services are always available. Effective performance testing of web applications is a critical task because it is not possible to know beforehand how many users will actually be connected to a real-world running application. Web application page consists of many file types for example HTML, JavaScript and images. Images represent the biggest component of the web pages (Muntean, McManis & Murphy, 2001). Some of the pages also have a large number of images. Both size and number of images could affect both network and server performance since images often take several seconds to load (Harrison, Dey & Hudson, 2010), especially in peak hours when there are a lot of clients visiting the page. This dissertation report is the generation of various test cases by using Loadrunner in three scenarios with different sizes of images in order to analyse the factors that affected the web performance and find out how images influenced the access time of the web pages.

1.3 Project Objectives

The objectives of this research are summarized as below:

(i) To develop web application with three different scenarios based on the size of images.

(ii) To design test cases on performance testing for the three scenarios in (i).

(iii) To execute test cases on Loadrunner.

(iv) To compare the results using performance parameters which are the successful request rates and response time in web application in (i) by using Loadrunner.

1.4 Scope of Project

This study focused on the analysis of performance testing in web application testing. The Loadrunner testing tool by Hewlett-Packard (HP) was used to test the successful request rates and response time in web applications. The web application domains which were the three scenarios with different sizes of images were analysed and compared by using Loadrunner testing tool based on the parameters.

1.5 Significant of Study

In the software development life cycle, testing is highly needed to assure the quality of the software process and product. Performance testing is critical for the success of the web application. One of the major bottlenecks in performance is the limited bandwidth of the network that connects browsers to the servers. Hence, if the amount of data flowing through the network can be reduced, it is possible to improve the response times and support more users using same network infrastructure. The main aim of this research is to prove that the influence of image size on the web performance. This provides the opportunity to improve application's performance before it becomes available for demanding users.

1.6 Expected Outcomes

The main aim of this research is to prove that the influence of image size on the web performance. The basic criterion of comparison are successful request rates and response time. Three scenarios of online shopping system were developed with different sizes of images. The three scenarios with different sizes of images were analysed and compared by using Loadrunner testing tool based on the parameters. This research outcome is to show how the size of images affect the web application performance.

1.7 Outline of the Dissertation Report

This dissertation report consists of five chapters. Chapter 1 is used to explain main objectives of the dissertation report. It consists of background of study, problem statement, the scope of works covered, the objectives of the dissertation, significant of study and expected outcomes. Chapter 2 illustrates the literature review of Loadrunner testing tool and brief explanation on the software testing for web application. Chapter 3 discusses the methodology and tool in a way to obtain the entire objectives of this dissertation report. Chapter 4 explains the implementation and detailed steps used in this work. Chapter 5 discusses the objectives achievement, disadvantages, future work and the conclusion.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter discusses on the literature of software testing and performance testing. It begins with a cursory review of software testing such as the concept and definition of software testing, the testing technique and the general classification of software testing techniques. Next, it presents the web application testing. This is followed by the definition of functional testing and non-functional testing. Finally, it discusses a few related work on the performance of web application testing and ends with a summary.

2.2 Software Testing

Testing is a procedure of investigating a system or its components with the expectation to determine whether it fulfills the defined prerequisites or not. This activity will bring out the real, expected and the difference in their outcomes. Basically testing is executing a system in order to identify missing features or errors which are not accounted for in the prerequisite (Pratibha & Manju, 2014). As per ANSI/IEEE 1059 standard (IEEE, 1994), testing can be characterised as a procedure of investigating a software application to discover the differences between the existing and desired conditions such as the errors, bugs, and defects in order to assess the characteristics of the software application.

.

2.3 Types of Testing

This section describes the different types of testing which may be used on the system under test (SUT). There are two types of testings; manual testing and automated testing. This dissertation report based upon automated testing.

2.3.1 Manual Testing

This kind of testing incorporates the physical testing of the software without utilising any automated tools or any script. In manual testing, the tester assumes the role of an end user and tests the software to recognise any unexpected features or bug (Pratibha & Manju, 2014). There are several distinctive stages for manual testing such as Unit Testing, Integration Testing, System Testing and User Acceptance Testing.

Testers will use test plan, trials or test stimulation to diagnose the software to confirm that a complete testing is done. Moreover, manual testing consists of exploratory testing as well since the testers need to explore the software before being able to identify errors.

2.3.2 Automated Testing

Automation testing which is also called Test automation is used when the tester composes scripts and utilises an alternate programming to test the software (Pandey, 2013). This methodology includes automation of a manual procedure. Automation test is utilised to re-run the test situations which perform manually, rapidly and repeatedly. Besides regression testing, automation testing is additionally used to test the application for load, performance and stress. Compared to manual testing, automation test expands the test scope, enhances accuracy, saves money and time (Angmo & Sharma, 2014).

It is impossible to computerise all the features in the software. Nonetheless the areas whereby large number of the users access the software at the same time to perform exchanges for instance registration forms, login form and so forth should be automated (Pandey, 2013).

Besides, all Graphical User Interface (GUI) products, associations with databases, field validations and so on can be effectively tested via computerising process and then manual procedure. Test automation ought to be used by considering the following for the software (Pereira, 2012):

(i) Large and critical projects.
(ii) Projects that require testing the same areas frequently.
(iii) Requirements do not change frequently.
(iv) Access the application for load and performance with many virtual users.
(v) Stable software with respect to manual testing.
(vi) Availability of time.

By considering the above situations, the user will be able to determine and decide on when to use the test automation to solve the problems for the software. Based on these situations, it is obvious that automation does not really solve the problems all the time. Despite of this, it is known that there are certain situations where automation is proved to be very helpful. To clarify more on this, automation has many advantages in regression testing, static and repetitive tests, load and performance testing, data driven testing and smoke testing (Fernandes & Fonzo, 2014).

Automation testing is used in order to reduce the testing time of complex activities such as regression testing and other extensive or laborious testing process that are involved in the software development cycle. Automation testing is commonly used for the projects that require regular testing of the same sections of code at any time of the development stage.

Automation test is carried out by utilising a supportive programming language such as Visual Basic (VB) scripting and a computerise software applications. Several tools which are readily available can be utilised to write automation scripts. Before identify the tools, it is important to distinguish the methodology that can be utilized to automate the testing first (Pandey, 2013).

Identifying areas within a software for automation is hard and not cost-effective to automate the whole software testing process. This is mainly due to the expensive nature of the testing tools. Not only this, it may also be due to the unstable nature of certain section of the codes used. Thus it is important to identify the areas that need to automate the testing first.

(i) Selection of appropriate tool for test automation

There are a few types of testing tools available but it is very crucial to keep in mind of the testing nature that involved when the right tools are being chosen for the automation.

(ii) Writing test scripts

Test cases and scripts need to be developed to cover the software's large sections which might not able to be fully covered by an individual. Writing the test scripts is also to ensure that the large sections of the software are properly functioning.

(iii) Development of test suites

In order to ensure that the automated test cases run one after another without any manual intervention, the development of test suites are being carried out. To develop the test suites, multiple test cases, command line tool and a library are created in order to run the test suite.

(iv) Execution of scripts

The execution of test scripts can be done either manually with the option of test suites to run being chosen by the developer or automatically by using a monitoring mechanism as the substitute. Test scripts execution ensures that problems are identified effectively in case any other issue arises as the effect from code change or other factors.

(v) Create result reports

Result reports format needs to be created so that the details of the actions performed during the testing from individual test logs can be recorded. Not only this, the type of test reports format to be created, messages, screenshots and many more will also need to be defined.

(vi) Identify any potential bug or performance issue

An individual needs to easily identify any problems during the testing and the factors that cause them. Apart from that, these individuals also need to find out the events of any test operations that have failed and set the problems right in order to gain greater testing efficiency.

Following are the tools which can be used for automation testing:

(a) HP Quick Test Professional – This software provides regression and functional automation test for software environments and applications.
(b) Selenium – This software provides a record and playback tool for authoring test without the need to learn test scripting language.
(c) IBM Rational Functional Tester – This software is a tool for automated testing of the software applications which allows the users to create tests that mimic the actions and assessments of a human tester.
(d) SilkTest – This is a tool for regression testing and automated function of enterprise applications.
(e) TestComplete – It is a functional automated testing platform which gives testers the ability to create automated tests for iOS, Android, Microsoft Window and Web applications.
(f) Testing Anywhere – This software allows the developers and the testers to test the applications, controls, Web sites, GUI front-ends and objects.
(g) WinRunner – This is an automated functional GUI testing tool which allowes the user to record and play back user interface (UI) interactions as the test scripts.

(h) LoadRunner – This software is an automated performance and test automation for application load testing by examining system behaviour and performance while the actual load is being generated.

(i) LoadUI – It is a load testing software that mainly targeting at the web services.

(j) Visual Studio Test Professional – This is an integrated toolset developed to facilitate a plan-test-track workflow for cooperation between developers and testers.

(k) WATIR – It stands for Web Application Testing in Ruby. This is an open-source family of Ruby libraries for automation of the web browsers.

Although there are a lot of tools for performance testing, Loadrunner testing tool has been chosen to be used in this dissertation report.

2.4 The Software Testing Strategy

There are different methods which can be used for software testing. This section briefly describes those methods.

2.4.1 Black-Box

Black-box testing includes testing the software for its usefulness, it is utilised to figure out the errors in the structure of the data, interface errors and faulty functions and so on. Figure 2.1 shows the black box Testing. This testing disregards inner system of a framework (Khan & Sadiq, 2011). It analyses bugs just as per programming malfunctioning as they are discovered in its error outputs. It is utilised to discover erroneous functions that prompt undesired outputs when performed and wrong conditions which yield incorrect outputs when they are performed. Techniques listed below utilised black box testing strategies to test a system:

(i) Boundary Value Analysis (BVA)

(ii) Robustness

(iii) Worst Case Scenario

(iv) Equivalence Partitioning

(v) Decision Table

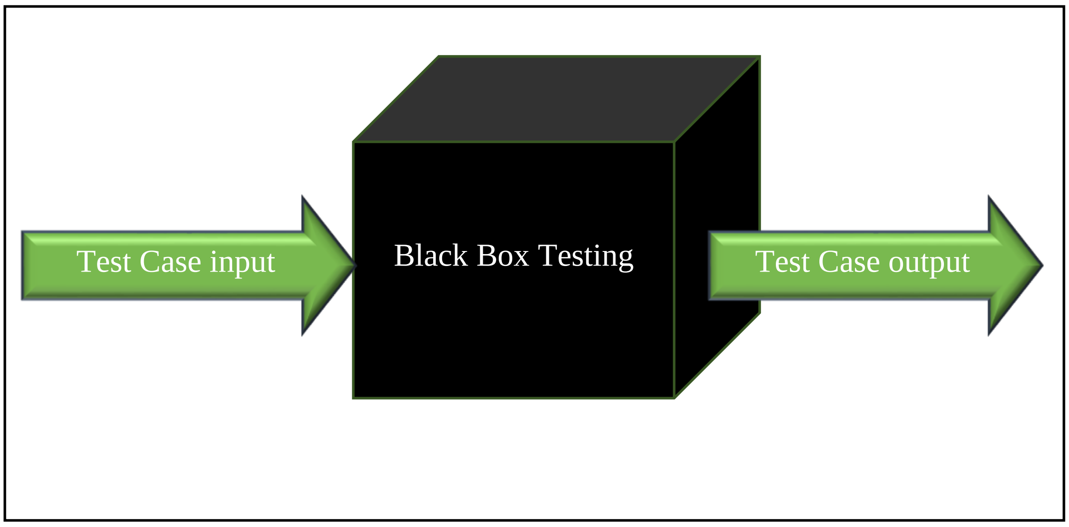

Figure 2.1: Black Box Testing

Black Box Testing permits us to complete most of the testing classes. Most of the testing can be exclusively done by using this testing method. In addition, Black box testing requires fewer resources (Khan & Sadiq, 2011).

2.4.2 White-Box

White-box testing considers the internal system of a framework or components. It is otherwise called as structural testing, clear box testing or glass box testing. It includes testing of all logic of a program, testing of loops, condition testing and data flow based testing. Figure 2.2 shows White Box Testing. This assists in recognising errors which occur even in incomplete and unclear programming specifications. The aim of this testing is to guarantee that the trials practice can be done in every path of a program. The trials likewise guarantee that all independent paths in a program have been utilised at very minimum once (Khan & Sadiq, 2011). All internal data structures are practiced to guarantee validity. All loops are executed to their limits and within operational bounds. Every branch is practiced at least once. By utilising white-box testing, a software engineer can plan trail tests which:-

(i) Exercise independent paths inside a module or unit.
(ii) Exercise logical choices on both their actual and false side.
(iii) Execute loops at their limits and within their operational bounds.
(iv) Practice with all internal data structures to guarantee their validity.

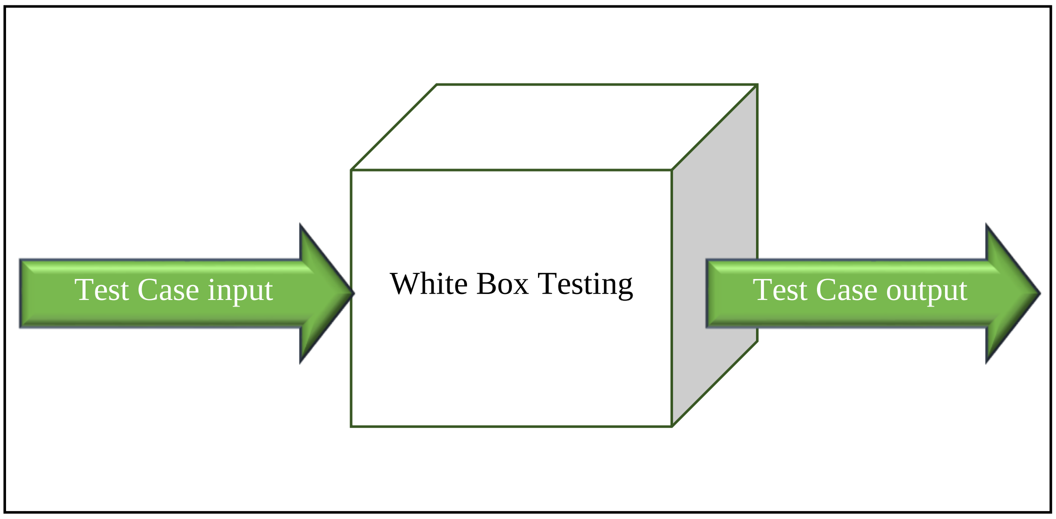

Figure 2.2: White Box Testing

White box testing can cover a bigger section of the program code while testing and it can disclose typographical errors as well. However, test trials need to be altered if usage changes (Khan & Sadiq, 2011).

2.4.3 Gray-Box

The third testing technique which is known as gray box testing has been considered as well. It is used for testing software which the tester has the knowledge of its basic code or logic. It is focuses around the internal data structures and algorithms for planning the trial tests more than black-box testing method yet less than white-box testing method. Figure 2.3 shows the Gray Box Testing. This technique is vital when conducting integration testing between two modules of code composed by two different programmers, whereby just the interfaces are exposed for the testing. Likewise, this technique can incorporate reverse engineering to fix limit values. Gray box testing is non-intrusive and fair on the grounds that it does not need the tester to know the source code (Jovanović & Irena, 2010).

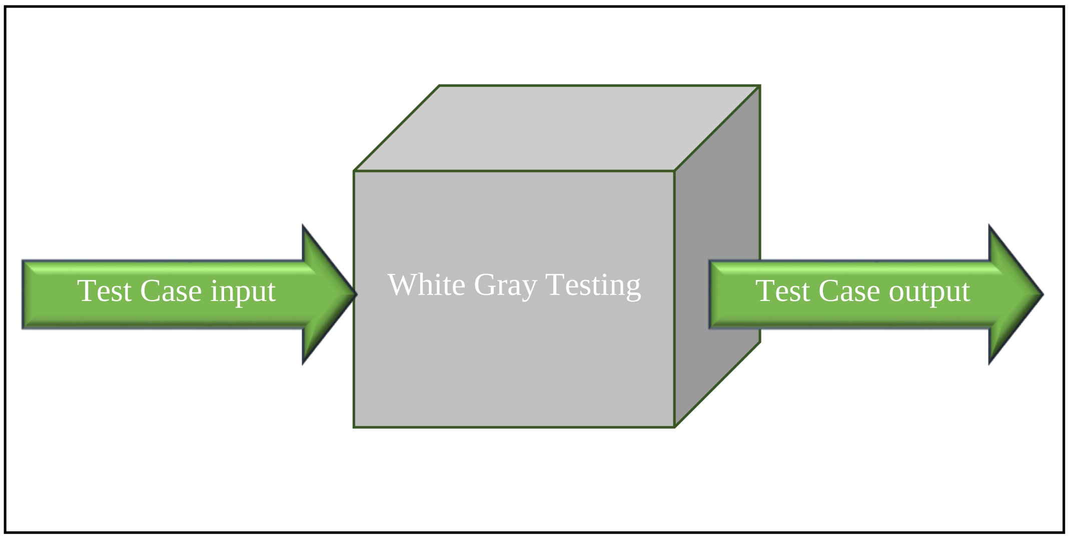

Figure 2.3: Gray Box Testing

Gray box testing can test software application by utilising powerful mix of both white- box testing and black- box testing technique. This is an effective and efficient method to test the application (Acharya & Pandya, 2012). The methodology use in this study, focused on black- box testing.

2.5 Web Application Testing

Web testing aims to do software testing that concentrates on web applications (Arora & Sinha, 2013). A complete web application testing needs to be done before publishing in order to trace any errors in web features. There are many tools in the market that can be used for testing the web interface and application (Rick, 2014). These tools are utilised for load, performance and stress testing of web sites, web applications, web servers and other web interfaces. Web testing is important for investigating bottlenecks and performance leakage in the site or web application being tested. Web testing is gaining importance in light of the fact that web applications are among the rapidly developing classes of programming systems being used today. These applications are generally used to aid an extensive variety of essential dealing for example business functions mainly product sale and distribution, scientific functions for example proposal review and data sharing and medical functions for example diagnoses based on expert system (Arora & Sinha, 2013).

2.6 Web Application Perspectives

There are different testing perspectives that have been proposed by numerous researchers over the years. Software testing divided into two distinct perspectives, non-functional testing to test how the service should perform describing the quality properties of the implementation of functional concerns. The second perspective is functional testing. It tested what the service should do (Schmeling, Charfi & Mezini, 2010). These two perspectives each held different software testing activities that were complementary to each other (Di, Giuseppe & Fasolino, 2006). Although there are different testing perspectives of web application, this dissertation has tested the performance of web application which is under testing the non-functional requirement of web application. Figure 2.4 shows the different testing perspectives which are explained in next subsections.

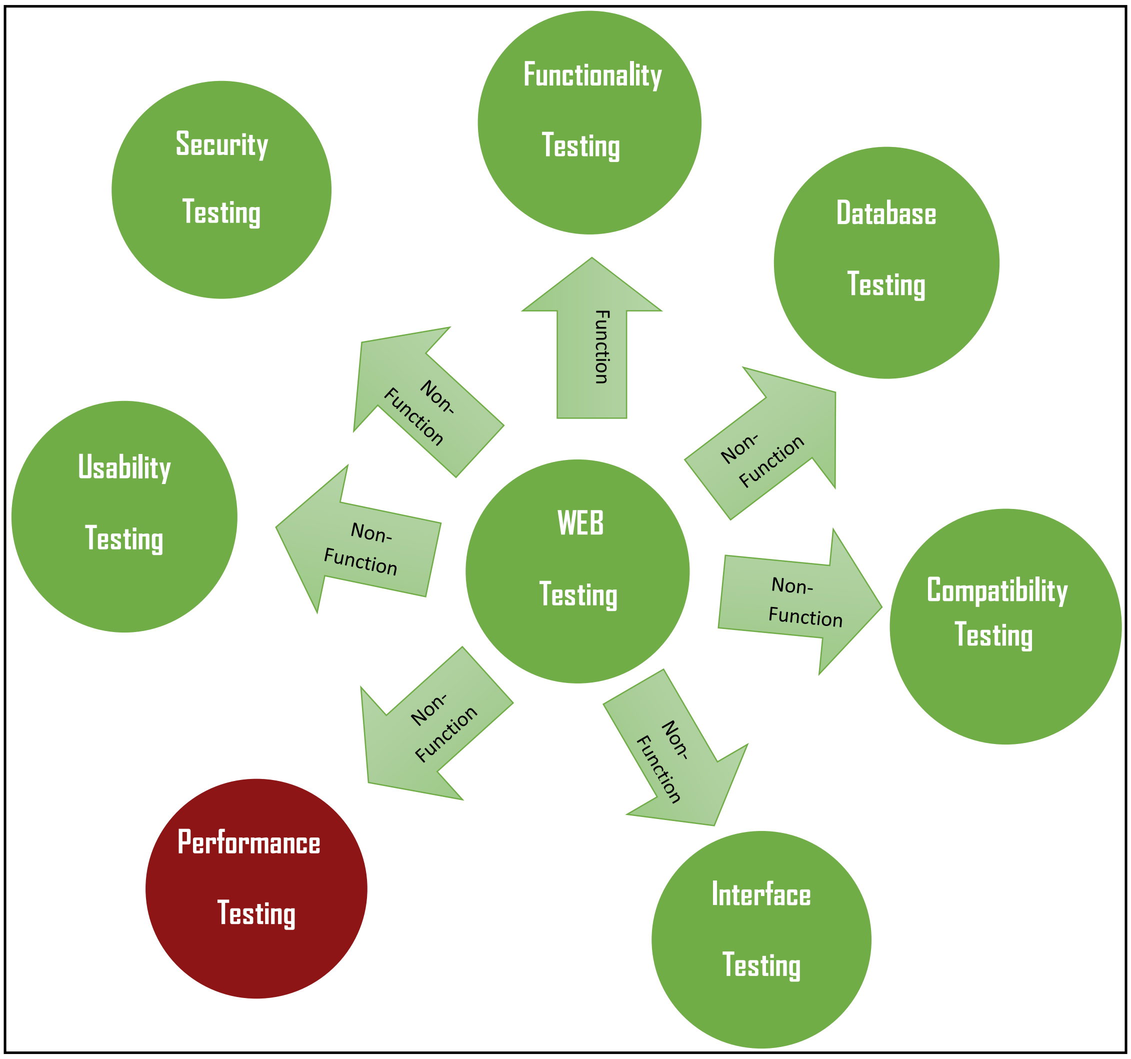

Figure 2.4: Types of Web Testing

2.6.1 Testing the Functional Requirement of Web Application

Testing the functional requirements of an application aims at verifying that an application’s features and operational behaviour correspond to their specifications (Di, Giuseppe & Fasolino, 2006). In other words, this type of testing is responsible for uncovering application failures due to the faults in the functional requirements’ implementation, rather than failures due to the application’s running environment. In order to achieve this aim, any failures due to the running environment should be avoided or reduced to a minimum.

2.6.2 Testing the Non-Functional Requirement of Web Application

There are different types of non-functional requirements that a web application either explicitly or implicitly is usually required to design with specific aims. Descriptions of the verification activities that can be executed to test the main non-functional requirement of a web application are presented in the following:

2.6.2.1 Usability Testing

Usability is the degree to which an item can be utilised by defined users to accomplish a particular objective or objective efficiently, effectively and satisfactorily. Usability is thought to be a standout amongst the most essential quality of web applications. Usability testing system measures the ease of use of a system’s user interface (UI) and distinguishes particular issues and concerns related with a particular kind of usability testing. To execute the usability testing successfully, there are five attributes that need to be addressed (Isa *et al*., 2014). The five attributes are:

(i) Set specific objectives for each test
(ii) Participants represent genuine clients
(iii) Participants do genuine task
(iv) The person conducting the research observes and records what participants do
(v) The researcher does data analysis, diagnose issues, and gives suggestions for changes

2.6.2.2 Compatibility Testing

Compatibility testing is a kind of software testing utilised to guarantee compatibility of the application, system or site fabricated with different objects for example, other web browsers, hardware platforms, users (in the event that a particular prerequisite is specified such as a user who can read and communicate in a specific language only), operating systems and so forth. This kind of testing assists the researcher to discover how well a framework performs in a specific environment that incorporates hardware, networking, operating systems and other programming tools and so on. It is essentially the testing of an application or an item constructed within the computing environment. Additionally, it tests whether the application or the software item constructed is compatible with the operating system, database or other programming systems and hardware (ISTQB, 2014).

2.6.2.3 Security Testing

An essential step in security assessment is to recognize the security threats and risks by considering presumed abilities of the attackers (Savola & Karppinen, 2007). In data security, a risk can be characterised by three variables which are:

(i) The likelihood of a threat which includes hazards to safety
(ii) The likelihood for vulnerabilities and the potential effect
(iii) The threats are possible at any time during the entire life cycle of the system being evaluated

Security threats in a system are not stagnant as security algorithms and solutions are discovered, new vulnerabilities will crop up each now and then (Savola & Karppinen, 2007). Security testing is identified by check the application security administrations and to distinguish potential security deformities. A complete web security testing ought to cover infrastructure, deployment, input validation, verification, approval, configuration, sensitive data management, encryption, session administration, working parameters, exemption administration, auditing and logging and a few other aspects (Qian *et al*., 2013). Doing risky investigation and development of security prerequisites are the vital part of security testing processes. Without legitimate necessities, it would be tricky to make a testing arrangement and accomplish genuine results. Web Security Testing goes for safety testing objectives proposed by an organisation's product improvement projects, summarising the fundamental kind of tests, and for a particular segment proposed test.

2.6.2.4 Database Testing

Database testing includes the tests to check the accurate qualities which have been recovered from the database by the web or desktop application. Data ought to match correctly with the records stored in the database. Database testing is one of the real testing which obliges the tester to have abilities in checking tables, composing queries and procedure. Testing can be performed in web application or desktop and database can be utilised as a part of the application for example Standardized Query Language (SQL) or Oracle. There are numerous ventures like financing, banking and healthcare protection which requires far extensive database testing (Mandeep, 2012).

2.6.2.5 Interface Testing

GUI testing is a methodology to test the user interface of an application and to check if application is functioning correctly (Kanchan & Madhuri, 2014). GUI testing includes carrying out the same set of tasks and comparing and contrasting the consequence of the same and the normal yield and reproduce same set of tasks at different times with diverse information and same level of accuracy. Besides, it incorporates how the application handles keyboards and mouse usage, how distinctive GUI parts like menu bars, toolbars, dialogues, buttons, edit fields, list controls, pictures and so forth respond to client data and whether it performs in a desired way. Using GUI testing for application in the early stages of the software improvement cycle accelerates the development, enhances quality and diminishes risk towards the end of the cycle. GUI testing can be performed both manually with a human tester or could be performed automatically by utilising a software program (Sandeep, 2003). Another type of non-functional testing is performance testing which has been chosen to be used in this dissertation report. Performance testing is described in details in section 2.7.

2.7 Performance Testing

Performance testing is one of the fundamental activities traversing the entire life cycle of software engineering. Configuring of test environment is an essential stage before testing web applications and the suitability of a test environment will significantly influence the authenticity and precision of the test outcomes. Presently web service procedures are generally utilised in the field of business incorporated information systems, and enterprise class business-to-business applications. Before large scale web services systems are launched on the Internet, their performances need to be evaluated. Testing machines need to be purchased to assemble the essential hardware test environment. Additionally complex software environment has to be configured. Therefore a lot of investment, including a lot of time is needed to configure test environment. Besides, web application testing has numerous challenges for example testing a large number of virtual users, testing from various geographical areas, managing simultaneous demands from users and developing test flexibly (Zhang *et al.*, 2011).

2.7.1 Performance Testing Types

Web performance testing is an imitation of end-users of the tested systems done by documenting and depicting the real user's behaviour using an automated controlled approach that repeats the implementation of the user's behaviour, since it is self–executing, the system will be able to stimulate high-traffic user's behaviour. Test systems mostly use the test generator to stimulate the user's activities (Zhu, Fu & Li, 2010). Figure 2.5 shows the Schematic Diagram for Performance Testing.

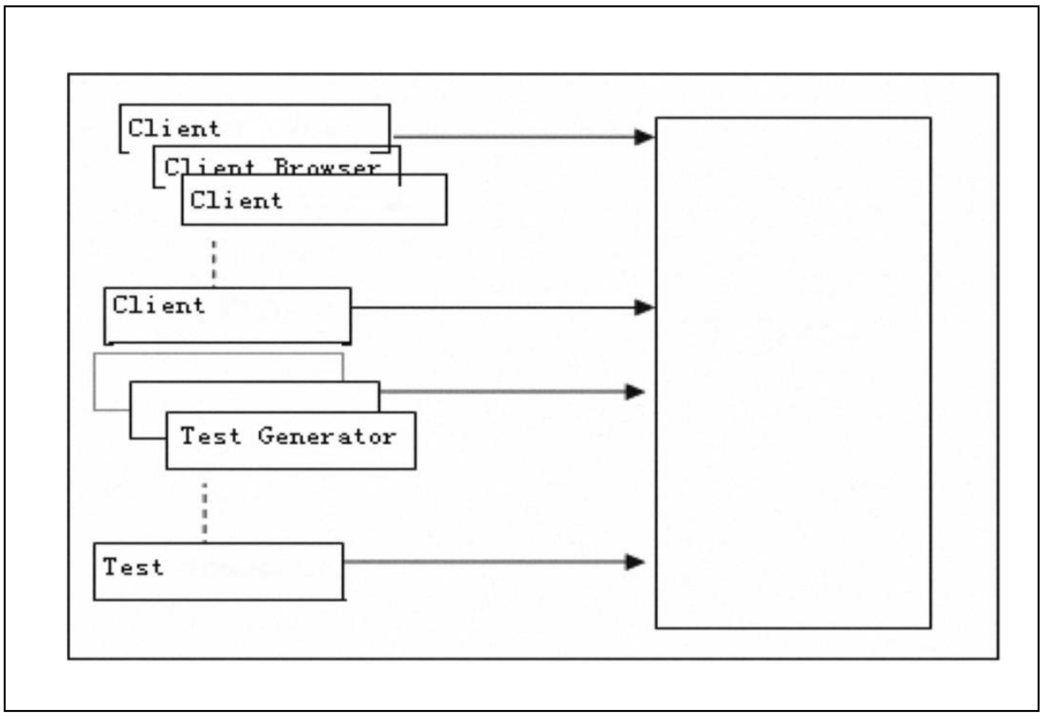

Figure 2.5: Schematic Diagram for Performance Testing (Zhu, Fu & Li, 2010)

Normally, a test generator can stimulate user's behaviour and run hundreds of web customer software. Communication between the virtual users and web servers can be done directly without using web browsers for example Internet Explorer or Firefox. During the performance tests, the running and testing the number of the virtual users can be specified in the generator. In the event where more virtual users need to be stimulated, it can be linked to multiple load generators, and a centralised control system. In this way, a flow can be created closer to the limit set. Moreover, delay time can likewise be set between the implementation of the two tests. There are three types of web performance testing namely stress testing, load testing and strength test. Each test utilises the same testing tools, script and environment but varied testing time intervals.

2.7.1.1 Load Test

Loading test is carried out by steadily adding the load and checking the performance of the system and eventually fixing the maximum load the system can withstand, and at the same time fulfill the performance markers. Hence, the performance of a system when different work load are applied can be analysed by using load testing. The changes that occur in various system performance markers when the load is gradually increased can be ascertained. Load testing is done to depict a particular kind of stress testing that is done in order to enhance the number of users testing the application. At the same time, load testing is done by gradually increasing the number of users from relatively small till the application response time is over. Load testing and stress testing can be integrated (Zhu, Fu & Li, 2010).

2.7.1.2 Stress Test

Stress testing is carried out by continuously increasing the system load and checking the changes in system performance, and eventually determining the performance of the system under any different conditions till it reaches failure state. Therefore, the system can offer the largest number of service tests. By changing the input of the application which corresponds with the increasing load, the conditions under which the performance of the application will becomes intolerable can be identified. The disclosure of the inflection point in the performance of the application is done to distinguish the bottlenecks in a system and performance point that cannot be received so that the maximum service level test that a particular system can provide can be obtained. Stress tests analyse the current hardware and software environment which the system can withstand by identifying the maximum load and bottlenecks of the system (Zhu, Fu & Li, 2010).

2.7.1.3 Strength Test

Strength test is a stress test or a longer interval load test. Strength test is a different test from other forms of tests as the weight-bearing or tension testing interval of only ten per seconds to maintain the strength test needs to be deferred a couple of hours or even days. Strength testing often detects some peculiar errors for instance memory leaks namely memories, rollback fragments which exist in the database transactions which are not submitted, or have a cumulative influence on system errors, resources and others (Zhu, Fu & Li, 2010).

2.7.2 Performance Testing Criteria

Performance testing which is done using automated testing tools which reenact a variety of normal and abnormal peak load conditions of the indicators is used for performance testing of a system. Performance model stipulates measurable standards of performance. The standard comprises of a series of performance indicators. Typical performance metrics are system throughput, response time, system resource utilisation, network Traffic Statistics, HTTP transactions/sec and the number of session per second and the number of concurrent users, resource request queue length and other markers which measure web performance (Zhu, Fu & Li, 2010). The following discourse concentrates on the initial three pointers.

2.7.2.1 Response Time

Response time is a time characterised from the start of a request made by a user till the final respond from the server. Response time is the key software performance of a web application. Hence, page response time is denoted by network time (N1+N2+N3+N4) and application time (A1+A2+A3) as shown in Figure 2.6.

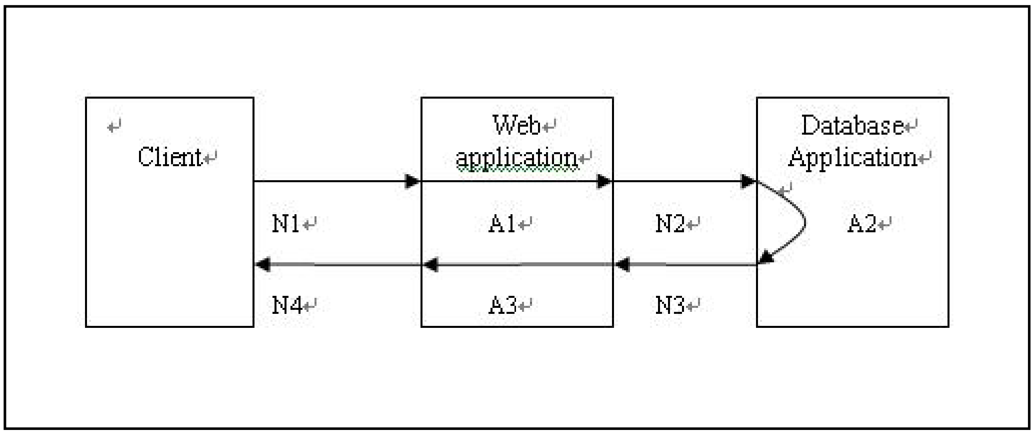

Figure 2.6: Web Application Pages Response Time (Yunming & Xu, 2009)

2.7.2.2 Concurrency User

During testing, the testing engineer concentrates on business concurrency users that is how many concurrency users from the business model are present. In equation (2), the C is the mean of the concurrency users, N is the number of login session, L is the mean length of the login session and T is the reviewed time (Yunming & Xu, 2009). A login session denotes an interval from start to the end of a session.

$$C = \frac{N*L}{T} \tag{1}$$

$$\mathrm{Cp} = \mathrm{C} + 3 * \sqrt{C} \tag{2}$$

In equation (2), if the login session is fit to Poisson distribution, Cp is crest concurrency users. For instance, in a system, there are 3000 users and the average number of users who visit this system every day are 400 users. For a typical application, the mean time begins when the user logs into the system and ends when the user log out after around 4 hours' time, a user normally works 8 hours a day.

In this way, as indicated by equation (1) and (2), the C and Cp are:

$$C = \frac{400*4}{8} = 200$$

$$\mathrm{Cp} = 200 + 3 * \sqrt{C} = 242$$

2.7.2.3 Average Response Time

In order to evaluate website performance and user's feeling more accurate and complete, there is a special performance index which is average response time. Based on Equation (3) (Boonchieng, 2014), the average response time are calculated by dividing total Response Time with the number of the response.

$$\text{Average Response Time} = \frac{\sum \text{Response Time}}{n} \tag{3}$$

2.7.2.4 Successful Request Rate

In order to evaluate website performance and user's feeling more accurate and complete, there is a special performance index which is successful request rates. Based on the Equation (4) (Guangzhu & Shujuan, 2009), successful request rates are calculated by dividing total successful request of Online Shopping System with Total request test cases. The results are the final result for the successful request rates of the performance Testing.

$$\text{Successful request rates} = \frac{\sum \text{number of success requests}}{\text{number of total requests}} \tag{4}$$

2.7.3 Performance Testing Tool

When conducting tests on web applications there are various performance testing tools that can be used to test the web application. This research is focused on the Loadrunner testing tool. HP Loadrunner is an automated performance and test automation product from Hewlett-Packard for application load testing. It is testing system behaviour and performance, while generating actual load. A software testing tool, HP Loadrunner works by creating virtual users who take the place of real users' operating client software, such as Internet Explorer, sending requests using the HTTP protocol to IIS or Apache web servers. HP Loadrunner can simulate thousands of concurrent users to put the application through the rigors of real-life user loads, while collecting information from key infrastructure components (Web servers, database servers etc.) The results can then be analysed in detail to explore the reasons for particular behaviour. HP Loadrunner also supports various protocol bundles for load testing: .NET Record/Replay, Database, distributed component object model, GUI Virtual Users, Java Record/replay, network, oracle e-business, remote access, remote desktop, rich internet applications, Systems Applications and Products (SAP), Service-Oriented Architecture (SOA), web and multimedia and wireless.

REFERENCES

Angmo, R., & Sharma, M. (2014). Performance evaluation of web based automation testing tools. In confluence the next generation information technology summit (Confluence). International Conference, pp. 731-735.

Arora, A., & Sinha, M. (2013). Dynamic content testing of Web Application using user session based state testing. In *Confluence 2013: The Next Generation Information Technology Summit (4th International Conference),* pp. 22-28.

Acharya, S. & Pandya, V. (2012). Bridge between Black Box and White Box–Gray Box Testing Technique. International Journal of Electronics and Computer Science Engineering, 2(1), pp. 175-185.

Boonchieng, E. (2014). Performance and security issue on open source private cloud. In *Electrical Engineering Congress (iEECON), 2014 International*, pp. 1-5.

Chakrapani, A. & Ramesh, K. V. (2011). Automated Functional Testing Using IBM Rational Robot. International Journal of Computer Science and Information Technologies, Vol. 2 (3), pp. 977-981.

Di Lucca, Giuseppe. A., & Fasolino, A. R. (2006). Testing Web-based applications: The state of the art and future trends. *Information and Software Technology*, *48*(12), pp. 1172-1186.

Fernandes, J. & Fonzo, A.D. (2010).When to Automate Your Testing (and When Not To). Retrieved on December 20, 2014, from http://www.oracle.com /technetwork/topics/qa-testing/whatsnew/when-to-automate-testing-1-30 330.

Harrison, C., Dey, A. K., & Hudson, S. E. (2010). Evaluation of progressive image loading schemes. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1549-1552.

ISTQB (2014). What is Compatibility testing in software testing. Retrieved on November 04, 2014, from http://istqbexamcertification.com/what-is-compatibility-testing-in-software.

Isa, W. A. R. W. M., Lokman, A. M., Wahid, E. S. & Sulaiman, R. (2014). Usability testing research framework: Case of Handicraft Web-Based System. In *Information and Communication Technology (ICoICT), 2nd International Conference on*, pp. 199-204.

IEEE (1994). IEEE Guide for Software Verification and Validation Plans. USA: IEEE Std 1059.

Jovanović & Irena, (2010). Software Testing Methods and Techniques. The IPSI BgD Transactions on Internet Research. pp. 30-41.

Guangzhu & Shujuan. (2009). A quick testing model of Web performance based on testing flow and its application. In *Web Information Systems and Applications Conference. WISA. Sixth*, pp. 57-61.

Kanchan, G., & Madhuri, S. (2014). "An Approach to Generate the Test Cases for GUI Testing. International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 6, pp. 2348 – 7968.

Kalita, M., & Bezboruah, T. (2011). Investigation on performance testing and evaluation of PReWebD: a. NET technique for implementing web application. *Software, IET*, *5*(4), 357-365.

Križanić, J., Mošmondor, M., & Lazarevski, P. (2010). Load testing and performance monitoring tools in use with AJAX based web applications. In*MIPRO, 2010 Proceedings of the 33rd International Convention,* pp. 428-434.

Khan, M. A., & Sadiq, M. (2011). Analysis of black box software testing techniques: A case study. In *Current Trends in Information Technology (CTIT), 2011 International Conference and Workshop on,* pp. 1-5.

Krishnamurthy, B., & Wills, C. E. (2000). Analyzing factors that influence end-to-end Web performance. Computer Networks, 33(1), pp. 17-32.

Lu, Y., Wu, H., & Wang, Y. (2006). Web application performance analysis based on comprehensive load testing. In *Wireless, Mobile and Multimedia Networks, 2006 IET International Conference on*, pp. 1-4.

Mandeep, S. (2012). Database Testing Basics - How to test and what to test .Retrieved on November 11, 2014, from http://www.softwaretestingtimes.com /2012/01/database-testing-how-to-test-and-what.html.

Mădalina, M. L. A. K. (2007). Analyzing the Network Response Time and Load Balancing. Revista Informatica Economică, pp. 64-67.

Muntean, C. H., McManis, J., & Murphy, J. (2001). The influence of Web page images on the performance of Web servers. In *Networking—ICN 2001*, pp. 821-828.

Pratibha, F. & Manju, K (2014). Research of Load Testing and Result Based on Loadrunner. (SSRG-IJCE). ISSN: 2348-8352, pp. 1-4.

Pandey, K. A. (2013). Review on Software Testing Methodology. Journal of Engineering Research and Application Vol. 3, Issue 5, Sep-Oct 2013, pp. 579-583.

Pereira, j. (2012). Software Testing Fundamentals. Automation Testing. Retrieved on November 08, 2014, from https://www.linkedin.com/pulse/software-testing-fundamentals-joao-carlos-fitas-rosado-pereira.

Qian, L., Wan, J., Chen, L., & Chen, X. (2013). Complete Web Security Testing Methods and Recommendations. In *Computer Sciences and Applications (CSA), 2013 International Conference on*, pp. 86-89.

Rick, H. (2014). Web Site Test Tools and Site Management Tools. Retrieved on September 23, 2014, from http://www.softwareqatest.com/qatweb1.html

Schmeling, B., Charfi, A., & Mezini, M. (2010). Non-functional concerns in web services: requirements and state of the art analysis. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, pp. 67-74.

Savola, R., & Karppinen, K. (2007). Practical Security Testing of Telecommunications Software--A Case Study. In *Telecommunications. AICT 2007. The Third Advanced International Conference on*, pp. 7-7.

Sandeep (2003). GUI Testing. Retrieved on November 13, 2014, from http://www.appperfect.com/products/application-testing/app-test-gui-testing.html.

Varesh, T. (2012). Testing Process in Load Runner in Testing. Retrieved on November 08, 2014, from http://www.c-sharpcorner.com/UploadFile/face6d/load-runner-testing-process-in-testing/

Wu, Q., & Wang, Y. (2010). Performance testing and optimization of J2EE-based web applications. In Education Technology and Computer Science (ETCS), 2010 Second International Workshop on Vol. 2, pp. 681-683.

Yunming, P. & Xu, M. (2009). Load testing for web applications system. International Conference on Information Science and Engineering. pp. 2954-2957.

Zhang, L., Chen, Y., Tang, F., & Ao, X. (2011). Design and implementation of cloud-based performance testing system for web services. In *Communications and Networking in China (CHINACOM), 2011 6th International ICST Conference on*, pp. 875-880.

Zhu, K., Fu, J., & Li, Y. (2010). Research the performance testing and performance improvement strategy in web application. In *2010 2nd International Conference on Education Technology and Computer* (Vol. 2), pp. 328- 332.