Bachelor of Science in I.T. Management

Subject: **Final Year Project**

Project Title**:** **A Comparison of Web Testing Tools**

Project Supervisor: Margaret Finnegan

Student Name: Yemisi Adedeji

Student Number: X00068052

Contents

Abstract

Software testing in present era is the process of validating and verifying the correctness of software. Automated testing tool enables the developer and tester to automate the whole process of testing in software development life cycle (SDLC). Testing is very important phase of SDLC where the software is examined properly and modifications are proposed. Thus testing is necessary for quality of service provided by software. Web service is widely used concept now a days and less literature is available regarding web service performance and SOAP messaging.

The objective of this paper is to conduct the comparative study of automated tool for web services as a leading tool in black box test automation. This study will help in the promotion and usage of various open source web service tool toward performance of real time network using quality of service(QOS) provided by these tools. Further in this research paper the evaluation and comparison of three automated software testing tools is done to determine their usability and effectiveness.

Chapter 1: Introduction

The aim of this research is to evaluate and compare three web testing tools to determine their usability and effectiveness. Software testing is a crucial part of software development and the process of automating software testing is vital to its success. Currently, complex systems are being built and testing throughout the software development cycle is pertinent to the success of the software.

The success of web service technology is clearly evident from the usage and adoption of this IT technology. A large number of providers from different sectors of industry are shifting to web service technology. Web services are software components accessible through programmatic interfaces and can perform tasks from simple requests to complex processes (Zhang, D.S 2004). The heterogeneous nature of web service technology offers advantages like interoperability, usability, use of standardized communication protocol, deploy ability, etc. This makes web services technology an ideal candidate for organizations to host and deploy services in order to collaborate with other organizations in a flexible manner. In order to attain the trust of service users, it is necessary that the system must conform to the performance requirements as it is the most important criteria for evaluating a system (Hussain, S. et al 2013).

It is therefore necessary to test the system before deployment in order to ensure that the system meets quality of service requirements. Various testing tools have been developed and designed for testing of web services. By using these test tools, web engineers can perform their tasks easily and efficiently, thus improving the quality of the system.

Test tools automate the process of testing and are targeted to a specific test environment such as functional testing, performance testing, load testing, exception testing, etc. With the help of test tools, testers can create, manage and execute tests for a specific test environment for a particular application. The test results are compared with the expected results to evaluate the quality of the product. Web service testing is a quite challenging area for researchers. The importance of this can also be judged with the ongoing research in this field. Several methods and techniques proposed by researchers as well as development of testing tools. There are commercial as well as open-source test tools available today for testing of web services (Hussain, S. et al 2013).

To perform this research, a selected web application was first manually tested; then tested using the three automated testing tools. Automated testing included the development of scripts that not only saves time and resources when applications are updated, but also speeds up the process of testing when regression testing is necessary.

Chapter 2 Literature Review

2.1 Introduction

This section will review the academic literature and studies on the different types of software testing, the software testing tools for web services and web testing. It will also consider the ideas and techniques of software testing that have become essential knowledge for all software developers. These concepts presented in this section will be utilized in this research. Also in this section, a number of key terms will be defined and explained.

2.2 Software

Ammann, P. & Offutt, J. 2008 define software as a key ingredient in many of the devices and systems that pervade our society. Software defines the behaviour of network routers, financial networks, and telephone switching networks, the Web, and other infrastructure of modern life. Software is an essential component of embedded applications that control exotic applications such as airplanes, spaceships, and air traffic control systems, as well as mundane appliances such as watches, ovens, cars, DVD players, garage door openers, cell phones, and remote controllers. Modern households have over 50 processors, and some new cars have over 100; all of them running software that optimistic consumers assume will never fail! Although many factors affect the engineering of reliable software, including, of course, careful design and sound process management, testing is the primary method that the industry uses to evaluate software under development.

2.3 Test Automation for Web Applications

Many software applications today are written as web-based applications to be run in an Internet browser. The effectiveness of testing these applications varies widely among companies and organizations. In an era of highly interactive and responsive software processes where many organizations are using some form of agile methodology, test automation is frequently becoming a requirement for software projects. Test automation is often the answer. Test automation means using a software tool to run repeatable tests against the application to be tested. For regression testing this provides that responsiveness.

There are many advantages to test automation. Most are related to the repeatability of the tests and the speed at which the tests can be executed.

Test automation has specific advantages for improving the long-term efficiency of a software team's testing processes. Test automation supports:

- Frequent regression testing
- Rapid feedback to developers
- Virtually unlimited iterations of test case execution
- Support for Agile and extreme development methodologies
- Disciplined documentation of test cases
- Customized defect reporting
- Finding defects missed by manual testing

2.4 Software Testing

Binder, R.V. (2000) defines software testing as the execution of code using combinations of input and state selected to reveal bugs. Its role is limited purely to identifying bugs and diagnosing or correcting them (debugging). Bruegge, B. and Dutoit, A.H (2004) also mention that test inputs normally come from test cases, which specify the state of the code being tested and its environment, the test inputs or conditions, and the expected results. A test suite is a collection of test cases, typically related by a testing goal or implementation dependency. A test run is an execution of a test suite with its results. Coverage is the percentage of elements required by a test strategy that have been traversed by a given test suite. Regression testing occurs when tests are rerun to ensure that the system does not regress after a change. In other words, the system passes all the tests it did before the change.

Software Testing is a way to validate and verify the working of a particular product or application. It can be incorporated at various points of time in the development process depending upon the methodology and tools used. Testing usually starts after the crystallization of requirements. At a unit level, it starts concurrently with coding; whereas at an integration level, when coding is completed. Testing is used for finding out the bugs in our application. It helps in finding out the failure of the software before it crashes the application. The purpose is to satisfy the stakeholders and ensure the quality of an application with testing. Sohoni, G. (2014)

Meek, J. et al refers to software testing as an area of software development where persistence is essential. Software testing is the process of assessing software quality by using the software with applicable test cases to determine if proposed software requirements are being satisfied. Testing

applications thoroughly and efficiently is necessary for deployment when wanting to retain existing customers and also draw in new customers.

2.5 Software Testing Strategies

There are two main approaches to generating test cases: specification-based and implementation-based (or code-based). Specification-based testing, also known as black box or functional testing focuses on the input/output behaviour or functionality of a component (Crowther, D & Clarke, P. 2005). This technique treats the program under test as a „black box" where no knowledge about the implementation is assumed (Zhu, H. et al 1997). Code-based, also known as white box, generates a test suite based on the source code of the program (Poon, P.L et al 2010). This research paper is to perform specification-based testing on an application with three different tools and compare the way tests are conducted from all three and also compare the results.

2. 5. 1 Black Box

This type of testing looks at functionality of the application. Software can have different paths to get to a definite end path. Multiple representations of items also generally mean multiple execution paths. Two issues of complexity to consider are: the number of different execution states software can go through, and the issue of software concurrency. Software concurrency is sometimes used to make software faster or get done with its work sooner and concurrency is sometimes used to make software do more work over the same interval where speed is secondary to capacity (Silktest, n.d.). Concurrency is becoming more prevalent at many levels. Attempting to black box integrations of poor quality components (the traditional „big bang" technique) has always been ineffective, but large systems make it exponentially worse. Dependencies among units must be controlled to make integration quality truly feasible (Stobie, K. 2005).

Research has been found to show that there are building blocks for test guidance algorithms, the step evaluation, the state evaluation and the evaluation order (Kervinen, A., Virolainen, P. 2005). In black box testing, the coverage of the specified behaviour of an application is measured. Applications sometimes use third party components like COM (component object model) or CORBA (common object request broker architecture). When applications use these, software developers, testers, and users are not always aware of the complex relationships that are created when functionality comes from an external component. Since external components can have bugs, starting to observe data being passed to third party components and the return values of function

calls can reduce the time needed to track down application bugs Whittaker, J., Thompson, H. (2003).

2. 3. 2 White Box

Another technique in testing is white box testing, which is testing the code behind the software to ensure the requirements of the software are met. One way of creating an automated software testing tool is to create test cases for specific three-variable functions (Meek, J. et al 2011). Functional testing techniques are limited by the nature of the programming problem and the number of inputs into the system. Suitable programs for functional testing must have a relatively small number of independent inputs. Functional testing techniques allow suitable means for the systematic generation and validation for certain programming problems (Meek, J. et al 2011). White box testing includes creating test cases and then test suites to assist in running multiple test cases at once. If two test cases test the same features of a unit, if one test case is updated correctly and the other is not, one test may fail while the other passes. This makes the test results ambiguous (Koochakzadeh, N., Garousi, V. 2010).

2.6 Importance of Software Testing

Since software has become even more complex today, there are now more lines of code; and the more the lines of code the higher the probability of errors happening. This implies that more thorough testing needs to be done.

Testing is important because software reliability is defined using testing and approximately fifty percent of the software development budget for software projects is spent on testing (Crowther, D., Clarke, P, 2011).

Prusch, A et al also mentioned that because software testing can be labour intensive and expensive; there is therefore a need to reduce human testing. Pressman, R. (2005) also agrees that software testing is also necessary because errors are often introduced into software inadvertently as it is designed and constructed. While testing of individual systems is important, testing integrated systems is even more important.

Also, Askarunisa, A et al (2009) agree that given the repetitive and labour intensive nature of software testing, finding the appropriate tool support is imperative if software testing is to become a mature discipline.

The problem is that when modifications are made on applications, if there is no process in place to perform automatic testing on applications, then the time spent on testing may be too great. If

automatic scripts are written, it will save time and resources when applications are modified in the future and regression testing becomes necessary

2.7 Software Testing Tools

Several software testing tools are currently available and it can be assumed that the quality, maintainability, testability and stability of the software are improved by using testing tools

Modern approaches to the Software Development Life Cycle (SDLC) encourage an iterative approach while testing; so it is an ongoing activity that occurs throughout the life of the project. However, systems testing and integration testing are also done just before the software is deployed. It is therefore clear that testing permeates the entire SDLC. Testing is also important because software is not only pervasive but also often integrated within multiple systems.

Askarunisa, A et al (2009) believe that these tools assist software engineers in increasing the quality of the software by automating mechanical aspects of the software-testing task.

Once the test scripts are developed and associated test cases, they may be used repeatedly and save both time and resources. With the proliferation of software testing tools, it may be difficult to determine which automated software testing tool is best to use to achieve the goal of efficiently testing software.

2.8 Web Services

Web services are software components accessible through programmatic interfaces and can perform tasks from simple requests to complex processes (Zhang, D.S. 2004). The heterogeneous nature of web service technology offers advantages like interoperability, usability, use of standardized communication protocol, deployability, etc. This makes web services technology an ideal candidate for organizations to host and deploy services in order to collaborate with other organizations in a flexible manner.

Web services (sometimes called *application services*) are services (usually including some combination of programming and data, but possibly including human resources as well) that are made available from a business's Web server for Web users or other Web-connected programs. Providers of Web services are generally known as application service provider s. Web services range from such major services as storage management and customer relationship management (CRM) down to much more limited services such as the furnishing of a stock quote and the

checking of bids for an auction item. The accelerating creation and availability of these services is a major Web trend (Tech Target, n.d.)

2.9 Web Testing

Web testing is the name given to software testing that focuses on web applications. Complete testing of a web-based system before going live can help address issues before the system is revealed to the public. Issues such as the security of the web application, the basic functionality of the site, its accessibility to handicapped users and fully able users, its ability to adapt to the multitude of desktops, devices, and operating systems, as well as readiness for expected traffic and number of users and the ability to survive a massive spike in user traffic, both of which are related to load testing.

2.10 Terminology

It is important to have a basic understanding of terms used in this research. In the following section, the researcher defines terms used in this paper.

- **Application:** An application is any program, or group of programs, that is designed for the end user. (Beal, V. n.d.)
- **Applications software:** (also called end-user programs) includes database programs, word processors, and spreadsheets. Figuratively speaking, applications software sits on top of systems Software because it is unable to run without the operating system and system utilities. (Beal, V. n.d.)
- **Automated testing**: Automated software testing is a process in which software tools execute pre-scripted tests on a software application before it is released into production. The objective of automated testing is to simplify as much of the testing effort as possible with a minimum set of scripts. (Rouse, M. 2014)
- **Automated testing tool:** A device that provides a mechanical or mental advantage in automated testing.
- **Benefit:** To be useful or profitable to. In this study, the term refers to a positive value that is derived from utilizing the automated testing software. According to QA Labs, there are five key factors of quality in reference to Web Applications (QA Labs, 2000).
    - Quality

- Reliability
- Recoverability
- Security
- Usability
- Performance

- **Bug:** A bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result.
- **Business Rule:** A business rule describes the operations and constraints that apply to an organization in achieving its goals. They are often collected as part of the Requirement gathering process to be used in Use cases.
- **Business Analyst:** A person who is responsible for identifying the business needs of their clients and stakeholders to help determine solutions to business problems.
- **Defect:** A flaw in any aspect of the system including the requirements, the design or the code, that contributes, or may potentially contribute, to the occurrence of one or more failures.
- **Graphical User Interface (GUI):** A program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex command languages. On the other hand, many users find that they work more effectively with a command-driven interface, especially if they already know the command language.
- **Load Testing:** The practice of modelling the expected usage of a software program by simulating multiple users accessing the program's services. Concurrently.
- **Object Oriented:** A popular buzzword that can mean different things depending on how it is being used. Object-oriented programming (OOP) refers to a special type of programming that combines data structures with functions to create re-usable objects (see under object-oriented programming). Object-oriented graphics is the same as vector graphics.

  Otherwise, the term object-oriented is generally used to describe a system that deals primarily with different types of objects, and where the actions you can take depend on what type of object you are manipulating. For example, an object-oriented draw program might enable you to draw many types of objects, such as circles, rectangles, triangles, etc.

Applying the same action to each of these objects, however, would produce different results. If the action is Make 3D, for instance, the result would be a sphere, box, and pyramid, respectively.

- **Quality Assurance:** The process of ensuring that the quality of a product or process is sufficient to meet the needs of the stakeholders
- **Requirement:** A description of what a system should do. Systems may have from dozens to hundreds of requirements. The IEEE Standard Glossary of Software Engineering Terminology (1990) defines a requirement (Wiegers, K. 2003) as
    - A condition or capability needed by a user to solve a problem or achieve an objective.
    - A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
    - A documented representation of a condition or capability as in 1 or 2. (Wiegers, K. 2003)
- **Script:** A computer program that automates the sort of task that a user might otherwise do interactively at the keyboard.
- **Software Computer instructions or data:** Anything that can be stored electronically is software. The storage devices and display devices are hardware. Software is often divided into two categories: Systems software: Includes the operating system and all the utilities that enable the computer to function; and Applications software: Includes programs that do real work for users. For example, word processors, spreadsheets, and database management systems fall under the category of applications software.
- **Software Development Life Cycle:** A framework for describing the phases involved in developing information systems. The cycle consists of the following phases:
    - Analyse
    - Design
    - Implement
    - Test
    - Deliver
    - Support

Some popular models of a SDLC include the waterfall model, the spiral model, and the incremental build model.

- **Software Testing:** A process used to identify the correctness, completeness and quality of developed computer software. Actually, testing can never establish the correctness of computer software, as this can only be done by formal verification (and only when there is no mistake in the formal verification process). It can only find defects, not prove that there are none.
- **Software Testing Tool**: A device that provides a mechanical or mental advantage in software testing.
- **Strategy:** A long term plan of action designed to achieve a particular goal.
- **Test Automation:** The use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.
- **Testing:** An approach to verification that involves systematically executing software to detect defects.
- **Test case:** A set of conditions or variables under which a tester will determine if a requirement is partially or fully satisfied. It may take many test cases to determine that a requirement is fully satisfied. In order to fully test that all the requirements of an application are met, there must be at least one test case for each requirement unless a requirement has sub requirements. In that situation, each sub requirement must have at least one test case.
- **Test Engineer:** The test engineer is an information technology (IT) professional who is in charge of one or more technical test activities, including designing test inputs, producing test case values, running test scripts, analysing results, and reporting results to developers and managers. (Ammann, P. & Offutt, J. 2008)
- **Time to market:** The length of time it takes to get a product from idea to marketplace.
- **Tool:** A device that provides a mechanical or mental advantage in accomplishing a task.
- **Use Case:** A technique for capturing the potential requirements of a new system or software change. Each use case provides one or more scenarios that convey how the system should interact with the end user or another system to achieve a specific business goal.

- **Web Application:** An application delivered to users from a web server over a network such as the World Wide Web or an intranet. This type of application has two very important characteristics:
    - A Web application is designed from the start to run in a Web-based environment. This means that the hypermedia aspect in terms of hypertext and multimedia in combination with traditional application logic must be taken into account throughout the application lifecycle, which makes it different with respect to a conventional application.
    - A Web application is an application, not just a set of Web pages. In particular, this implies that it enforces the notion of a session, which differentiates it from the ordinary request-response Web paradigm. In this context, even a Web service that dynamically generates pages may not be considered a Web application. Think for example of a timetable service that, given desired departure and destination times and places, returns a set of pages containing the available trains and connections. In this case, there is no need for the service to maintain the notion of session, which means that, in our view, this is not a Web application, but just a Web-based service.
    - Feichtner, C. et al (2003)
- **Web Based Applications:** A Web-based application refers to any program that is accessed over a network connection using HTTP, rather than existing within a device's memory. Web-based applications often run inside a Web browser. However, Web-based applications also may be client-based, where a small part of the program is downloaded to a user's desktop, but processing is done over the Internet on an external server. Web-based applications are also known as Web apps. (Technopedia, n.d.)

Chapter 3 Web Testing Tools

There are a number of open-source web service testing tools available in the software world. Although the core functions of these tools are similar, they differ in functionality, features, usability and interoperability. A brief description of each of them is presented below (Ghahrai, A. 2011)

3.1.1 Apache JMeter – Load and Performance tester

The Apache JMeter™ application is open source software, a 100% pure Java application designed to load test functional behaviour and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions. The Apache JMeter is a pure Java desktop application designed to load test functional behaviour and measure performance. It may be used to test performance both on static and dynamic resources (files, Servlets, Perl scripts, Java Objects, Data Bases and Queries, FTP Servers and more). It can be used to simulate a heavy load on a server, network or object to test its strength or to analyse overall performance under different load types. Apache JMeter may be used to test performance both on static and dynamic resources (Web services (SOAP/REST), Web dynamic languages - PHP, Java, ASP.NET, Files, etc. -, Java Objects, Data Bases and Queries, FTP Servers and more). It can be used to simulate a heavy load on a server, group of servers, network or object to test its strength or to analyse overall performance under different load types. You can use it to make a graphical analysis of performance or to test your server/script/object behaviour under heavy concurrent load (Apache Software Foundation n.d.).

3.1.2 Grinder – Java Load Testing Framework

The Grinder is a Java load testing framework that makes it easy to run a distributed test using many load injector machines. Load test anything that has a Java API. This includes common cases such as HTTP web servers, SOAP and REST web services, and application servers (CORBA, RMI, JMS, EJBs), as well as custom protocols. Test scripts are written in Python, and can call out to arbitrary Java code, providing support for testing a large range of network protocols.

The Grinder comes with a mature plug-in for testing HTTP services, HTTP scripts can be recorded easily from a browser session.

3.1.3 Multi-Mechanize – web performance and load testing framework

Multi-Mechanize is an open source framework for web performance and load testing. It runs concurrent Python scripts to generate load (synthetic transactions) against a remote site or service. Multi-Mechanize is most commonly used for web performance and scalability testing, but can be used to generate workload against any remote API accessible from Python.

Test output reports are saved as HTML or JMeter-compatible XML. It allows you to run simultaneous python scripts to generate load (synthetic transactions) against a web site or web service. You programmatically create test scripts to simulate virtual user activity. Your scripts will then generate HTTP requests to intelligently navigate a web site or send requests to a web service.

3.1.4 EarlGrey - Open Source Test Automation Tool for iOS

EarlGrey is a native iOS UI automation test framework released by Google that enables you to write clear, concise tests. It integrates with Xcode's Test Navigator so you can run tests directly from Xcode or the command line. One advantage of using EarlGrey test automation framework is its synchronization features which automatically synchronizes with the UI and network requests. EarlGrey uses screenshot differential comparison (also known as 'screenshot diffs') to determine the visibility of UI elements before interacting with them. As a result, you can be certain that a user can see and interact with the UI that EarlGrey interacts with.

3.1.5 Selenium – Web app testing tool

Selenium is a suite of tools such as Selenium IDE, Selenium Remote Control and Selenium Grid to test the web application. Selenium IDE is an integrated development environment for Selenium scripts. It is implemented as a Firefox extension, and allows you to record, edit, and debug tests. It supports record and playback. Selenium is a set of different software tools each with a different approach to supporting test automation. Most Selenium QA Engineers focus on the one or two tools that most meet the needs of their project, however learning all the tools will give you many different options for approaching different test automation problems. The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating

UI elements and comparing expected test results against actual application behaviour. One of Selenium's key features is the support for executing one's tests on multiple browser platforms.

3.1.6 Capybara – Acceptance test framework for web applications

Capybara helps you test web applications by simulating how a real user would interact with your app. Capybara aims to simplify the process of integration testing Rack applications such as Rails, Sinatra or Merb. Capybara simulates how a real user would interact with a web application. It is agnostic about the driver running your tests and currently comes with Rack Test and Selenium support built in.

3.1.7 OpenSTA – Open Systems Testing Architecture

OpenSTA is a distributed software testing architecture designed around CORBA. OpenSTA supplies versatile Test development software that enables you to create and run Tests tailor-made for the environment you are assessing. The current toolset has the capability of performing scripted HTTP and HTTPS heavy load tests with performance measurements from Win32 platforms. Results and statistics are collected during test runs by a variety of automatic and user controlled mechanisms. These can include scripted timers, SNMP data, Windows Performance Monitor stats and HTTP results & timings. The applications that make up the current OpenSTA toolset were designed to be used by performance testing consultants or other technically proficient individuals. This means testing is performed using the record and replay metaphor common in most other similar commercially available toolsets. Recordings are made in the tester's own browser producing simple scripts that can be edited and controlled with a special high level scripting language. These scripted sessions can then be played back to simulate many users by a high performance load generation engine. Using this methodology, a user can generate realistic heavy loads simulating the activity of hundreds to thousands of virtual users.

3.1.8 Pylot – Performance & Scalability Testing of Web Services

Pylot is a free open source tool for testing performance and scalability of web services. It runs HTTP load tests, which are useful for capacity planning, benchmarking, analysis, and system tuning. Pylot generates concurrent load (HTTP Requests), verifies server responses, and

produces reports with metrics. Tests suites are executed and monitored from a GUI or shell/console. It supports HTTP and HTTPS. It is multi-threaded and generates real time stats. Response is verified with regular expressions. GUI and Console mode support available.

3.1.9 WebLoad – The best LoadRunner Alternative

The WebLOAD Open Source Load Generation Engine is an open source project sponsored by RadView Software. This project is intended for ISVs, SIs and software developers who need to integrate a professional load generation engine into their applications.

3.1.10 Webrat – Ruby Acceptance Testing for Web applications

Webrat helps to write expressive and robust acceptance tests for a Ruby web application. It supports multiple Ruby web frameworks like Rails, Merb and Sinatra. It also supports popular test frameworks like RSpec, Cucumber, Test Unit and Shoulda.

3.1.11 Windmill – Web Testing Tool

Windmill is a cross-domain, cross-platform and cross-browser testing software. It runs on Microsoft Windows, Mac OS X, and Linux. It supports all major browsers: IE, Firefox, Safari, Chrome and Opera. In all these browsers you can use sterling IDE. There is no need to additionally use Visual Studio or Eclipse, since tests for any of the mentioned browsers can be written directly in the Windmill IDE.

3.2 Choosing an Automated Testing Tool

On selecting testing tool, we should take in consideration the following issues:

1. **Technology Under test**

   It should be very easy to understand and analyse, because we know the application that is under test but do we aware on all its sub processes, applications, and those applications that already run in the background? In few cases we might find out more applications, or/and special functional cases that we should take them in consideration. Before buying the tool, we must make sure we are not creating any testing limitation.

2. **Scripting Language**

Let us say that all my team members are well familiar with Java language, for them to build the frameworks with java will be very fast, which will shorten the TTM time. However, do we need testing application with scripting capabilities at all? Today code-free tools are available and there is variety of them, the question on the scripting language is now much wider.

3. **Support**

   every testing tool need support and bug fixing like every program, not fixing its defect will be equal to testing limitation, and with time we are wiser on the companies that provide this support. Using open source tool will give you the option to support the tool by yourself or external team. The support we will need should be handy and fast. Not having the support in time, might mean not providing the automated test you promise on time.

4. **Intuitive and Easy to Use**

   Learning curve for the tool is a second grade issue but still we must take it in consideration. Especially if we don't have time to teach and understand the tool (like in many extreme process development)

5. **Command Line – API – UI**

   In many cases we desire to activate the test from centralize application that will archive the results and even show us statistics on the overall cycle run. Do we have integration possibilities? One solution is activating the test via command line, in some cases the tool designed to work with OLE strategy, which will enable the possibilities to integrate it by code. So, how do I plan to activate the test? We need to summarize all the using methods needed before even testing it. I call it UI Availabilities.

6. **Non-Intrusive tool**

   Intrusive or non-intrusive testing tool? How much our application influenced by the fact that the testing tool is there?

7. **Brand**

   A well-known brand has its advantages like: reliability, resources availabilities, but there are disadvantages as well, like: higher price, support problems, long waiting for desired features

8. **Cost**

   Cost is an issue like every other and even more, we cannot decide on tool, which is too expansive, so we must understand our budget limitations and strategy. The cost of the tool

is not always so obvious, start using specific tools means purchasing more than single, stand-alone license, working with testing tools will need licensing for production needs and development needs as well, we must consider maintenance cost, support, management and leads, all are cost derived from our purchasing decision and in some cases are varied according the tool we selected.

3.3 Features are required in an automated testing tool

Testing is a critical part of the software development process. There are a lot of different automated software testing tools currently on the market. Some of these tools can perform only specific kinds of testing and work only with specific languages, like, for example, Java application unit testing. Other products support a wide range of applications and offer more features and functionality. (Palani, G. S. 2011)

The most important features that any automated test tool should have are listed below:

- Record and Playback
- Scripting Languages
- IDE Integrations
- Unit Testing Support
- BDD/TDD Support
- Data-Driven Testing
- Source Control Management

3.4 Web application testing methodologies

New methods and tools emerge quickly in the web application testing arena. The methodology and tools selected depends on the characteristics of the application and the development parameters, such as language and software. The use case for the application can also have an influence.

3.4.1 Usability testing

For an application to be effective, the user interfaces should comply with standards. Follow globally accepted conventions wherever applicable. For example, color coding conventions indicate that red is used to stop a process and green to start one.

Usability testing plays a pivotal role with applications that are designed to make manual tasks easier. The applications should comply with accessibility standards. For example, the widely used Captcha code has an option for spelling for people who are visually challenged.

Keep the following guidelines in mind when undertaking usability testing:

- Ensure proper navigation between web pages.
- Be sure to have a site map.
- Use appropriate color combinations and best practices.
- Avoid over-crowded content.
- Practice user friendliness to all types of users, from novice to expert.
- Provision support for physically challenged people.

3.4.2 User acceptance testing

The objective of user acceptance testing is to make sure your application meets the expectations of the user. It ensures that the application is fit enough to be deployed and used effectively. The following are tips for user acceptance testing:

- Ensure browser compatibility.
- Make sure that mandatory fields are given data in forms.
- Check for time outs and field widths.
- Be sure that proper control is used to feed data. For example, when requesting gender information, use an option button.

Alpha and beta testing are the two types of user acceptance testing.

Alpha testing: Testing in a development environment by developers.

Beta testing: Testing in a deployment or client environment by end users.

Performance testing: Performance testing on web applications measures the performance under various scenarios.

Performance tests include:

- **Stress testing:** To determine the maximum performance limits of an application.
- **Scalability testing:** To find out how adaptable the application is to changes in software and hardware.
- **Load testing:** To get an idea of how the application behaves under a heavy load. This test yields information and details about memory usage, CPU usage, and so forth.

- **Security testing:** Security testing for your application is very important if data leaks or modifications are unacceptable and intolerable. For example, if a university application includes the academic grades of a student, then security testing should ensure that the system cannot be hacked. For e-commerce applications, which sometimes involve banking transactions, security testing is critical. It should also ensure that sufficient authentication and authorization mechanisms are in place. Security testing can be static or dynamic.
    - **Static:** Static testing involves doing a static code analysis to check for any vulnerabilities. The goal is to understand the code flow and check for security threats by walking through the code.
    - **Dynamic:** Dynamic testing entails running the application to see if the response is as expected for the associated request. It is very similar to black box testing.

3.4.3 Functional testing

Functional testing ensures that individual functions are working well. Test cases should ensure that boundary conditions are tested. Invalid inputs should prompt appropriate error messages. In web applications, functional testing can range from testing whether links are working to checking whether changes made by users in a web page are reflected in the database. Some of the functional tests for web applications include:

- Database testing
- Configuration testing
- Compatibility testing
- Flow testing

3.4.4 Interface testing

Conduct interface testing to ensure that individual components are connected properly. The output of one module should be fed to the intended module without any issues. Interface testing plays a vital role for your applications that are developed to work on multiple platforms. The following are considerations to keep in mind during interface testing:

- Ensure that data flow occurs smoothly and as expected between modules in a single application and between applications.

- Ensure that the interfaces exposed by components are generic and extensible. They should be able to accommodate changes to the components while remaining backward compatible.

3.5 Criteria for Selection

Support for the testing process: This refers to wow a testing tool supports the actual testing process.

For each of the following items, we note whether they are supported by each of the testing tools. Possible result values for each criterion are: Y (for Yes), N (No), and P (Partial).

1. Support for creation of stubs/drivers/harnesses.
    - Does the testing tool provide a mechanism for generating stubs?
    - Does the testing tool provide a mechanism for generating drivers?
    - Does the testing tool provide a mechanism for generating test harnesses?

2. Support for comparing test results with an oracle.
3. Does the tool provide a mechanism for automatically comparing the results of the test against the expected results?
4. Support for documenting test cases.
5. Does the tool assist in keeping documentation of test cases?
6. Support for regression testing.
7. Does the tool provide the ability for automated regression testing?

Generalization of test information. What additional information can be provided from a testing tool, to provide further support for the testing process?

1. Support for the creation of test cases.
    - Does the tool provide help with creating test cases?
    - Does the tool provide help with the generation of implementation-based test cases?
    - Does the tool provide help with the generation of specification-based test cases?
2. Support for structural coverage.
    - Does the tool provide a measure of statement coverage?
    - Does the tool provide measure of branch coverage?

- Does the tool provide a measure of all-uses coverage?

Usability of testing tool. How easily can a given tool be installed, configured, and used? As these criteria are more subjective, these factors are rated for each of the selected tools. Each element listed here is given a ranking from 1 to 5, where 5 is always the positive or more desired result, 3 is an average or ordinary result, and 1 indicates a poor or very negative result. For example, "Learning Curve" is based on the time needed to become familiar with the tool before being able to use it in practice, and a shorter amount of time results in a higher score.

1. Ease of Use
    - a. How easy is it to install the tool?
    - b. How user friendly is the interface?
    - c. What is the quality of online help?
    - d. How helpful are the error messages?
2. Learning Curve
3. How long should it take an average programmer to be able to use the tool efficiently in practice?
4. Support
    - a. How quickly can you receive technical support information?
    - b. How helpful is the technical support provided?

Chapter 4 Methodology for Comparison

The overall goal of this research is to evaluate and compare several web testing tools to determine their usability and effectiveness. To accomplish this goal this research will download and run three free web testing tools, analyse and compare them, and produce some output based on the comparison. The research method includes the following:

1. Research and select a set of tools to be evaluated
2. Develop a metric suite to be used to evaluate the tools
3. Select target application to be tested
4. Document time spent manually testing the application and record data
5. Perform a feature assessment for the tools with the goal of ranking the tools based on their features and identifying the ideal feature set for an optimal tool
6. Test the target applications using the selected automated testing tools and gather resulting data
7. Evaluate and interpret results and the tools
8. Draw inferences and make recommendations

4.1 Selected Tools

The testing tools chosen in the comparison for stand-alone based testing were the Selenium – Web app testing tool, Windmill – Web Testing Tool and Pylot – Performance & Scalability Testing of Web Services.

4.1.1 Selenium Web App Testing Tool:

Selenium is a set of different software tools each with a different approach to supporting test automation. Most Selenium QA Engineers focus on the one or two tools that most meet the needs of their project, however learning all the tools will give you many different options for approaching different test automation problems. The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behaviour. One of Selenium's key features is the support for executing one's tests on multiple browser platforms.

Selenium is composed of multiple software tools. Each has a specific role.

Selenium 2 (aka. Selenium WebDriver)

Selenium 2 is the future direction of the project and the newest addition to the Selenium toolkit. This brand new automation tool provides all sorts of awesome features, including a more cohesive and object oriented API as well as an answer to the limitations of the old implementation.

Selenium 2.0 is the product of that effort. It supports the WebDriver API and underlying technology, along with the Selenium 1 technology underneath the WebDriver API for maximum flexibility in porting your tests. In addition, Selenium 2 still runs Selenium 1's Selenium RC interface for backwards compatibility.

Selenium 1 (aka. Selenium RC or Remote Control)

The Selenium Project, Selenium RC was the main Selenium project for a long time, before the WebDriver/Selenium merge brought up Selenium 2, the newest and more powerful tool.

Selenium IDE

Selenium IDE (Integrated Development Environment) is a prototyping tool for building test scripts. It is a Firefox plugin and provides an easy-to-use interface for developing automated tests. Selenium IDE has a recording feature, which records user actions as they are performed and then exports them as a reusable script in one of many programming languages that can be later executed.

Selenium-Grid

Selenium-Grid allows the Selenium RC solution to scale for large test suites and for test suites that must be run in multiple environments. Selenium Grid allows you to run your tests in parallel, that is, different tests can be run at the same time on different remote machines. This has two advantages. First, if you have a large test suite, or a slow-running test suite, you can boost its performance substantially by using Selenium Grid to divide your test suite to run different tests at the same time using those different machines. Also, if you must run your test suite on multiple environments you can have different remote machines supporting and running your tests in them at the same time. In each case Selenium Grid greatly improves the time it takes to run your suite by making use of parallel processing.

4.1.2 Windmill – Web Testing Tool

Windmill is a web testing framework that provides complete testing automation and strong debugging capabilities. It becomes more and more popular among the developers and QA testers because it aims at making test writing easier, portable and sustainable. With this in mind Windmill provides a robust architecture that allows flexibility and back-and-forth communication.

Windmill is a cross-domain, cross-platform and cross-browser testing software. It runs on Microsoft Windows, Mac OS X, and Linux. It supports all major browsers: IE, Firefox, Safari, Chrome and Opera. In all these browsers you can use sterling IDE. There is no need to additionally use Visual Studio or Eclipse, since tests for any of the mentioned browsers can be written directly in the Windmill IDE.

Windmill itself was developed using Python and JavaScript. Tests can be written in Python, JavaScript, Ruby, and there is a complete set of JSUnit functions. Moreover, Windmill provides a cross-browser test recorder that allows writing tests without learning a programming language. Simply record, edit, playback and interact with your tests from one interface.

Windmill advantages

Windmill includes wxWindmill Service UI that loads/runs test files and test directories. It can clear the test queue, launch Firefox, Safari and Internet Explorer, as well as kill the open Windmill test browser instances. Main Windmill features are:

- built-in Python shell that interacts with Windmill server
- built-in debugging tools: Firebug и Firebug Lite
- rich set of commands for interaction with the application
- DOM Explorer, XPath Explorer and Assertion Explorer integrated
- test saving, recording and playback
- SSL connection support
- reports generation on test running and performance
- malleable proxy API
- PDB debugging support

Windmill can compete with the currently existing automation tools. For example, Selenium doesn’t provide proxy manipulation or integrated debugging tools like Windmill. Selenium has wider browser and language support (C#, Java, JavaScript, Objective-C, Perl, PHP, Python, Ruby,

etc.), but Windmill has test recorder for all platforms, while Selenium provides plugin only for Firefox. Also Windmill is fully cross-domain.

4.1.3 Pylot – Performance & Scalability Testing of Web Services

Pylot is a free open source tool for testing performance and scalability of web services. It runs HTTP load tests, which are useful for capacity planning, benchmarking, analysis, and system tuning. Pylot generates concurrent load (HTTP Requests), verifies server responses, and produces reports with metrics. Tests suites are executed and monitored from a GUI or shell/console.
Developers, Testers, and Performance Engineers that need to test and tune the performance and scalability of their web services. It can also be used by Python programmers for integration into larger test suites. It requires familiarity with HTTP, XML, and performance testing to use this tool successfully.

Features:

- HTTP and HTTPS (SSL) support
- multi-threaded load generator
- automatic cookie handling
- response verification with regular expressions
- execution/monitoring console
- real-time stats
- results report with graphs
- custom timers
- GUI mode
- shell/console modes
- cross-platform

4.2 Evaluation Matrix

Minimum and maximum response time of testing tools for web services									
Tool	Web Service ID	Response Time (ms)							
		12:00AM		6:00AM		12:00PM		6:00PM	
		Min	Max	Min	Max	Min	Max	Min	Max
Selenium	W1	600	605	611	616	605	620	602	605
	W2	611	616	605	620	610	615	605	611
	W3	605	620	610	615	600	605	600	604
Windmill	W1	900	905	908	910	903	906	908	915
	W2	908	910	903	906	908	915	900	905
	W3	903	906	908	915	900	905	908	915
Pylot	W1	1192	1195	1202	1205	1190	1198	1210	1215
	W2	1190	1198	1210	1215	1210	1220	1210	1220
	W3	1210	1220	1210	1220	1210	1215	1190	1198

4.3 Comparative Study

Each testing tool had its own unique features and functions for testing; nevertheless, all tools had in common the test case. The test case is the most basic unit involved in testing, yet all other aspects of testing depend on it. Some tools require you to write the code for the test cases, while other tools generate test cases for you automatically.

4.4 Test Results

Each tool was examined according to the criteria presented in Table 1 above

Table 1 shows comparison of selected tools on the basis of application support, programming Language, OS support, license etc.

No.	Tool Name	Application Support	Programming language / Framework	OS Support	Browser	Developer	Website
1	Selenium	Web services /Web applications	C#/Java/ Haskell/ Java Script/Objective-C/Perl/PHP/Python/Ruby	Cross Platform	IE, Firefox, Safari, Chrome and Opera	seleniumhq	http://www.seleniumhq.org/

2	Windmill	Web services	C#/Java/ Haskell/ Java Script/Objective-C/Perl/PHP/Python/Ruby	Cross Platform	IE, Firefox, Safari, Chrome and Opera	github	https://github.com/windmill
3	Pylot	Web services	Python/wxPython	Cross Platform	IE, Firefox, Safari, Chrome and Opera	Corey Goldberg	http://www.pylot.org/

TABLE 1: ANALYSIS OF SELECTED TOOLS ON THE BASIS OF PLATFORM, VERSION AND USAGES

Table 2 shows the version detail of selected tools such as release date, version used etc.

Sr. No.	**Tool Name**	**1st Release Date**	**1st Version**	**Latest Release Date**	**Latest Version**	**Used Version**
1	Selenium	22/10/2004	1.0	30/06/2016	2.53.1	2.53.1
2	Windmill	12/06/2007	0.1.2	13/01/2011	1.5.0	1.5.0
3	Pylot	17/10/2007	0.1	06/07/2009	1.26	1.26

TABLE 2: VERSION'S DETAIL OF THE SELECTED TOOLS.

Comparative Study of the Selected Tools This section represents the comparison of four open source web service testing tools and two freeware web service testing tools along with their observed results. The observed results will help the researcher to determine the efficiency of suitable test tool for their needs. Temperature conversion web service is used to compare the selected test tools.

A. System Requirements:

All the test cases were run on an Intel Core i5 2.30 GHz processor machine with 4GB RAM, Microsoft Windows 8 Professional, and 2mbps Internet connection. The comparison is made between six tools with the input of same web service i.e. the temperature conversion from Celsius to Fahrenheit. Testing of the tools requires configuration which in turn involve installation, test environment setup, collection of data, analytical survey and selection of parameter. The sample web service i.e. temperature conversion is tested on the respective configure tools.

B. Approach Followed:

The tests were performed at the same instance of time and at same network speed. Based upon input test cases can be categorized into types that is valid test cases and invalid test cases. The critical parameters (response time and bytes processed) were evaluated to identify the performance of the testing tools. The observed results were analysed to determine the efficiency of the tool. Table 3 shows the response time of testing tools for valid input (Celsius,"100"), Table 4 shows the response time of testing tools for invalid input (Celsius," abc").

TABLE 3: RESPONSE TIME OF TESTING TOOLS FOR VALID INPUT (Celsius,"100")

Sr. No.	Tool Name	Input in Celsius	Output in Fahrenheit	Response Time	Bytes/sec
1	Selenium	100	212	609	409
2	Windmill	100	212	907	407
3	Pylot	100	212	1206	374

4.5 Analysis of Tools

From the results it is evident that each tool had its own architecture and internal processes which form the basis of comparative study of tools in terms of response time and bytes processed by test. The response time observed for various tools is shown in Table 3 & 4. Observed results, shows that Pylot takes maximum time to give response to web service than all other tools. Windmill is next while Selenium has the best response time which is about half the time it takes for Pylot to respond.

Hence from the values observed in all tables, it's clear that Pylot takes minimum response time for testing selected web service. The behaviour shown by Selenium with respect to response time clearly showed that it is the fastest tool amongst all selected tools. In open source web service tools Selenium took minimum time as output. Also the results of test cases are summarized to calculate average response time of each tool for web service i.e. temperature conversion.

This is represented in Table 5 below

Web Service Name	Average response time in ms		
	Selenium	Windmill	Pylot
Temperature Conversion	609	907	1206

Table 5: AVERAGE RESPONSE TIME OF TESTING TOOLS

It can be analysed from the table that the average response time for Selenium is better than other tools which are used for observation. In open source tools Selenium outperforms the rest of the open source tools. The observed data can also be represented in a graph which is shown in figure 1.

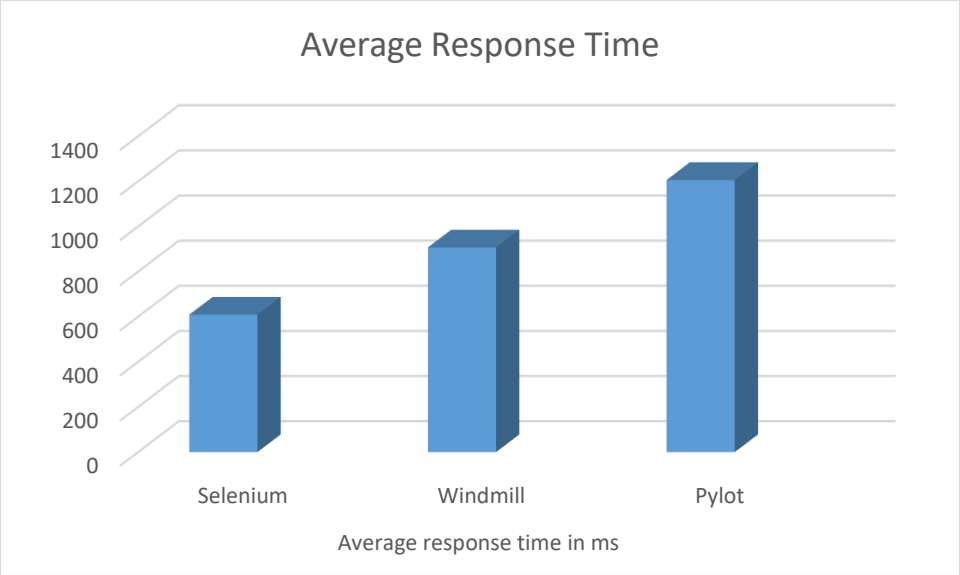

Figure 1: Average Response Time

Chapter 5 Conclusion

Web service technology is turning out to be the latest trend providing a new model of web. The rapid growth of web service market necessitated developing of testing methodologies, hence different methods and different tools proposed to test web services.

Testing a web service is challenging activity that involves many characteristics such as response time, throughput and calculation of bytes processed etc. The experimental approach used in this paper is based on real service implementation to retrieve and store data.

In this paper, the researcher presented a comparative study of open source web service testing tools with technical overview and features. Comparison was made on several quality factors including response time, throughput, and usability. Tools were evaluated by collecting the sample web services and collecting the test results. The comparison may give researchers an informative overview with potential benefits of open-source testing tools, and also help in promotion and development of open-source testing tools.

References

Ammann, P. & Offutt, J. (2008) Introduction to Software Testing Cambridge University Press

Apache Software Foundation (n.d.) Apache Jmeter (online) available at: http://jmeter.apache.org/ accessed July 7, 2016

Askarunisa, A et al (2009) N Ramraj, Shanmuga Priya, Selecting Effective Coverage Testing Tool Based on Metrics, The Icfai University Journal of Computer Sciences, Vol III, No. 3, 2009.

Beal, V. (n.d) application (application software) (online) available at: http://www.webopedia.com/TERM/A/application.html accessed 23rd May 2016

Binder, R.V. 2000 Testing Object-Oriented Systems. Addison-Wesley.

D. S. Zhang, "Web services composition for process management in E-business", Journal of Computer Information Systems, Vol. 45, No. 2, 2004, pp. 83-91.

Bruegge, B. and Dutoit, A.H. (2004) Object-Oriented Software Engineering Using UML, Patterns, and Java. Pearson Prentice Hall, 2004.

Crowther, D & Clarke, P. (2005), Examining Software Testing Tools, Dr. Dobb‟s Journal:Software Tools for the Professional Programmer, ISSN# 1044789X, Academic Search Premier, June 2005, Vol. 30, Issue 6.

Feichtner, C. et al (2003) Ubiquitous Web Application Development -A Framework for Understanding International Journal of Web Engineering and Technology, January 2003.

Ghahrai, A. (2011) Web Testing Tools (online) available at: http://www.testingexcellence.com/10-best-open-source-web-testing-tool/ accessed 23rd May 2016

Hussain, S. et al (2013) Web Service Testing Tools: A Comparative Study (online) available at: https://arxiv.org/ftp/arxiv/papers/1306/1306.4063.pdf accessed 23rd May 2016

Kervinen, A., Virolainen, P. (2005) Heurisitics for Faster Error Detection with Automated Black Box Testing, Science Direct, Electronic Notes in Theoretical Computer Science, Vol 111, January 2005

Koochakzadeh, N., Garousi, V. (2010) A Tester-Assisted Methodology for Test Redundancy Detection, Advances in Software Engineering, Hindawi Publishing Corporation, V 2010

Meek, J. et al (2011) Algorithmic Design and Implementation of an Automated Testing tool 54-59, Eighth International Conference on Information Technology: New Generations, 2011.
Palani, G. S. (2011) Summary of web application testing methodologies and tools (online) available at: http://www.ibm.com/developerworks/library/wa-webapptesting/ accessed 23rd May 2016
Poon, P.L et al (2010) A Framework for Specification-based Testing, ACM, Vol 53, No. 4, April 2010.
Prusch, A. et al (2011), Integrating Technology to Improve Medication Administration, AM J Health-Syst Pharm, Vol 68, ACM May 1, 2011.

Pressman, R. (2005) Software Engineering, A Practitioner's Approach, Sixth Edition, 2005.
Rouse, M. (2014) Automated software testing (online) available at: http://searchitoperations.techtarget.com/definition/automated-software-testing accessed 23rd May 2016

Silktest website, Silk Test (online) available at: http://www.borland.com/us/products/silk/silktest/ accessed 07 Jul. 16

Sohoni, G. (2014) Comparison of Automated Testing Tools: Coded UI Test, Selenium and QTP (online) available at http://www.dotnetcurry.com/tools/1004/comparing-automated-testing-tools-codedui-qtp-selenium accessed 25th April 2016
Stobie, K. (2005) Too Darned Big To Test, ACM Digital Library, Queue – Quality Assurance, February 2005, Vol 3, Issue 1.

(Tech Target, n.d.) Web services (application services) (online) available at: http://searchsoa.techtarget.com/definition/Web-services accessed 12th June 2016

Technopedia, (n.d.) Web-Based Application (online) available at https://www.techopedia.com/definition/26002/web-based-application accessed 12th June 2016

Whittaker, J., Thompson, H. (2003) Black Box Debugging, ACM Digital Library, December/January 2003-2004,Vol 3, Issue 1.

Wiegers, K. (2003) Software Requirements, Second Edition, Microsoft Press

Zhu, H. et al (1997) Software Unit Test Coverage and Adequacy, ACM Digital Library, ACM Computing Surveys, December 1997, Vol. 29, No. 4, page 371. URL: http://laser.cs.umass.edu/courses/cs521-621/papers/ZhuHallMay.pdf.