IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 3, March 2016.
www.ijiset.com
ISSN 2348 – 7968

# Analysis Of Code Coverage Through Gui Test Automation And Back End Test Automation

**Mr Tarik Sheth[1], Ms. Priyanka Bugade[2] , Ms. Sneha, Pokharkar[3]**
AMET University[1], Thakur College of Science and Commerce[2,3]

## ABSTRACT

Software testing provides a means to reduce errors, cut maintenance and overall software costs. Testing has become most important parameter in the case of software development lifecycle (SDLC). Testing automation tools enables developers and testers to easily automate the entire process of testing in software development. It is to examine & modify source code. The objective of the paper is to conduct a comparative study of automated tools such as available in market in Selenium and cucumber test tool. The aim of this research paper is to evaluate and compare automated software testing tools to determine their usability and effectiveness. There is wide variety of software testing tools available in market. Software testing tools has major features likes: web testing, window application etc.

**Keywords - SDLC, GUI Testing, Back end automation testing, Behavioral Driven Development. This topic covers the knowledge of software testing tools such as cucumber and selenium using automation testing and how the integration of selenium web driver is done using cucumber. It also tells us about the code coverage through GUI automation and back end automation testing**

## I. INTRODUCTION

While testing a software it has to go through various phases.Testing a particular software is not an easy task.Manually it takes much time to test, then by comparatively doing automation testing.The software has to follow the software development process, also known as a software development life cycle (SDLC),which is a structure imposed on the development of a software product. Software testing refers to process of evaluating the software with intention to find out error in it. Software testing is a technique aimed at evaluating an attribute or capability of a program or product and determining that it meets its quality. Software testing is also used to test the software for other software quality factors like reliability, usability, integrity, security, capability, efficiency, portability, maintainability, compatibility etc [1]. The aim of software testing process is to identify all the defects existing in a software product. Testing is the measurement of software quality. We measure how closely we have achieved quality by testing the relevant factors such as correctness, reliability, usability, maintainability, reusability and testability. Software is not unlike other physical processes where inputs are received and outputs are produced [2].The topic, Analysis of code coverage measurement through GUI automation and back end automation testing of the software covers all aspects of testing, a particular website or a web application..The purpose of this project is to invent our own test tool which will give more sophisticated outcomes then the **cucumber** tool which will be using .The outcome of our research tool should be more better then the testing tool which is already available in the market that is cucumber tool.. [3]. The paper tries to investigate and evaluate the effect of automation testing such as GUI and back end testing. [4].

The problems with manual testing are, it is very time consuming process, not reusable, has no scripting facility, great effort required, and some errors remain uncovered [5]. Automation testing covers all the problems of manual testing.

## II SIGNIFICANCE OF STUDY

The study of a cucumber tool will tell us what a testing tool is all about and how it reacts to the testing of a particular web application and how automation is carried out. As Cucumber is a tool based on Behavior Driven Development (BDD) framework which is used to write acceptance tests for web application. It allows automation of functional validation in easily readable and understandable format (like plain English) to Business Analysts, Developers, Testers, etc.Cucumber feature files can serve as a good document for all. There are many other tools like JBehave which also support BDD framework. Initially Cucumber was implemented in Ruby and then extended to Java framework. Both the tools support native JUnit.

Behavior Driven Development is extension of Test Driven Development and it is used to test the system rather than testing the particular piece of code.

## III RESEARCH METHODOLOGY

This is research work is use to evaluate and understand the working of testing tools and their efficiency and limitness.The survey based methodology is to analyse
That how a testing tool is more beneficial in the best way
And to see whether it meets the requirements of a various testing application. Because of the more advantages of the automation testing various companies are engaged in developing various automated test tools for various applications. There are two types of test tools. Open source

ISSN 2348 – 7968

test tools and Commercial test tools. Open Source Test tools- These test tools are free for the users to use. It can be downloaded from the internet or can be obtained by the vendor without any charges e.g. Selenium, test tools

*a. Overview of methods*

## Selenium

Selenium is one of the most popular automated testing suites. Selenium is designed in a way to support and encourage automation testing of functional aspects of web based applications and a wide range of browsers and platforms. Due to its existence in the open source community, it has become one of the most accepted tools amongst the testing professionals**.** Selenium supports a broad range of browsers, technologies and platforms.

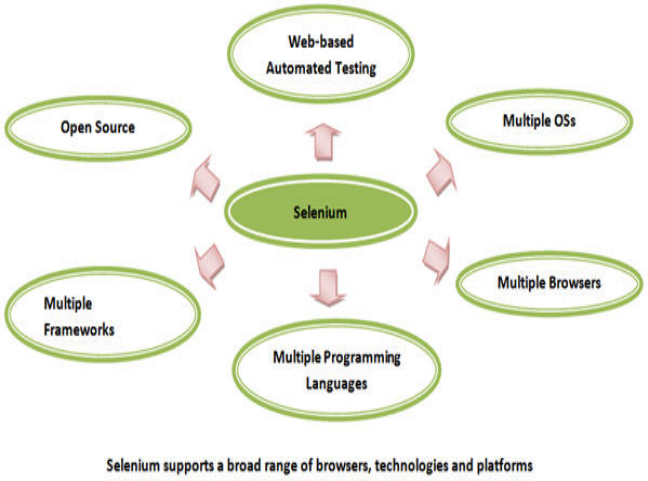



**Fig 1: Broad range of selenium tool**

Selenium is not just a single tool or a utility, rather a package of several testing tools and for the same reason it is referred to as a Suite. Each of these tools is designed to cater different testing and test.

**The suite package constitutes of the following sets of tools:**

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (RC)
- Selenium WebDriver
- Selenium Grid

## Selenium web drivers

Selenium Web Driver was created by an engineer named as Simon Stewart in the year 2006. Web Driver is also a web-based testing tool. Since, the tool was built on the fundamental where an isolated client was created for each of the web browser; no JavaScript Heavy lifting was required. This led to a compatibility analysis between Selenium RC and Web Driver. The selection of particular automated testing tool is based on the type of application we are testing and the cost associated with the tool. In the present work, we have evaluated the open source software testing tool Selenium.

```
1
2  package stepDefinition;
3
4  import java.util.concurrent.TimeUnit;
13
14 public class Test_Steps {
15     public static WebDriver driver;
16     @Given("^User is on Home Page$")
17     public void user_is_on_Home_Page() throws Throwable {
18         driver = new FirefoxDriver();
19         driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
20         driver.get("http://www.store.demoqa.com");
21         }
22
23     @When("^User Navigate to LogIn Page$")
24     public void user_Navigate_to_LogIn_Page() throws Throwable {
25         driver.findElement(By.xpath(".//*[@id='account']/a")).click();
26         }
27
28     @When("^User enters UserName and Password$")
29     public void user_enters_UserName_and_Password() throws Throwable {
30         driver.findElement(By.id("log")).sendKeys("sneha");
31         driver.findElement(By.id("pwd")).sendKeys("priyanka");
32         driver.findElement(By.id("login")).click();
33         }
34
35     @Then("^Message displayed Login Successfully$")
36     public void message_displayed_Login_Successfully() throws Throwable {
37         System.out.println("Login Successfully");
38     }
39
40     @When("^User LogOut from the Application$")
41     public void user_LogOut_from_the_Application() throws Throwable {
42         driver.findElement (By.xpath(".//*[@id='account_logout']/a")).click();
43     }
```

## Cucumber

**Cucumber** is a software tool that computer programmers use for testing other software. It runs automated acceptance tests written in a behavior-driven development (BDD) style. Cucumber is written in the Ruby programming language. Cucumber projects are available for other platforms beyond Ruby. Some use Ruby Cucumber with a bridge into the target language . Others use the Gherkin parser but implement everything else in the target language.[8] Cucumber allows the execution of feature documentation written in business-facing text.

Cucumber can be used along with Selenium, Watir, and Capybara etc. Cucumber supports many other languages like Perl, PHP, Python, .Net etc. In this tutorial we will concentrate on Cucumber with Java as a language.

In order to understand cucumber we need to know all the features of cucumber and its usage.

**#1) Feature Files:** Feature files are essential part of cucumber which is used to write test automation steps or acceptance tests. This can be used as live document. The steps are the application specification. All the feature files end with .feature extension.

ISSN 2348 – 7968

LogIn_test.feature | SeleniumTest.java | Test_Steps.java | TestRunner.java

```
1
2  Feature: Login Action
3
4  Scenario: Successful Login with Valid Credentials
5      Given User is on Home Page
6      When User Navigate to LogIn Page
7      And User enters UserName and Password
8      Then Message displayed Login Successfully
9
10 Scenario: Successful LogOut
11     When User LogOut from the Application
12     Then Message displayed LogOut Successfully
```

**Sample feature file:**
**Feature**: Login Functionality Feature
In order to ensure Login Functionality works,
To run the cucumber test to verify it is working

**Scenario**: Login Functionality
**Given** user navigates to SOFTWARETETINGHELP.COM
**When** user logs in using Username as “USER” and Password “PASSWORD”
**Then** login should be successful

**Scenario**: Login Functionality
**Given** user navigates to SOFTWARETETINGHELP.COM
**When** user logs in using Username as “USER1” and Password “PASSWORD1”
**Then** error message should be thrown

**#2) Feature:**
**T**his gives information about the high level business functionality (Refer to previous example) and the purpose of Application under test. Everybody should be able to understand the intent of feature file by reading the first Feature step. This part is basically kept brief.

**#3) Scenario:**
Basically a scenario represents a particular functionality which is under test. By seeing the scenario user should be able to understand the intent behind the scenario and what the test is all about. Each scenario should follow given, when and then format. This language is called as **“gherkin”.**

1. **Given:** As mentioned above, given specifies the pre-conditions. It is basically a known state.
2. **When**: This is used when some action is to be performed. As in above example we have seen when the user tries to log in using username and password, it becomes an action**.**
3. **Then:** The expected outcome or result should be placed here. For Instance: verify the login is successful, successful page navigation.
4. **Background:** Whenever any step is required to perform in each scenario then those steps needs to be placed in Background. For Instance: If user needs to clear database before each scenario then those steps can be put in background.
5. **And**: And is used to combine two or more same type of action.

*Junit Runner*
To run the specific feature file cucumber uses standard Junit JUnit Runner. It is a unit testing framework for the Java programming language. JUnit has been important in the development of testdriven development, and is one of a family of unit testing frameworks .

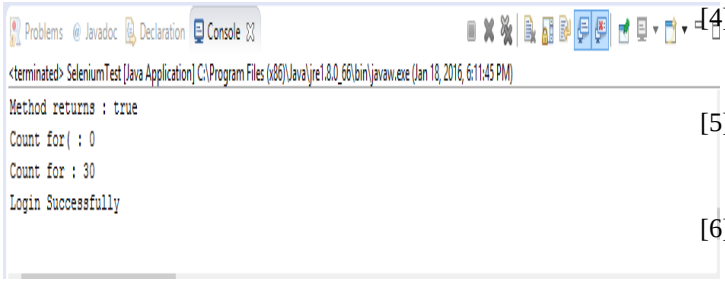
For Setting Up Selenium WebDriver along with cucumber test tool following steps are considered(in short):

- Set Up Java
- Set Up Eclipse
- Set Up WebDriver Client
- Configure Eclipse with WebDriver

## IV RESULT

```
driver.get("file:///C:/Users/Sneha/Desktop/new.html");

driver.get("file:///C:/Users/Sneha/Desktop/new.html");
String o = driver.getPageSource();

System.out.println(o);
String f  = "while(";
String b  = "for";
boolean retval = true;
int count=0,countfor=0,countb = 0,countforb=0;
int lastIndex = 0;
System.out.println("reached 1");

//for "for(" counting
 int index = o.indexOf(f);
 while(index != -1) {
     System.out.println(index);
     countfor = countfor +1;
     index = o.indexOf(f, index + 1);
 }


 //for "for" counting
 int index1 = o.indexOf(b);
 while(index1 != -1) {
     System.out.println(index1);
     countb = countb +1;
     index1 = o.indexOf(b, index1 + 1);
 }
```

ISSN 2348 – 7968




## V EVALUATION STUDY

There are a number of open source web testing and window application tools available in the software market. Although the core functions of these tools are similar, they differ in functionality, features, usability. Keeping in view its functionalities, we have selected the cucumber testing tool. For this study we use the current version of cucumber and selenium. Compariing these tools our new test tool will be made on the basis of these parameters :

- RECORDING EFFICIENCY
- USER INTERFACE TESTING
- TEST RESULT REPORTS
- REUSABILITY
- EXECUTION SPEED

## VI CONCLUSION

One can select a testing tool based on the type of application need to be tested, budget, and the efficiency required.The testing tool should show advantageous outcomes then the cucumber test tool.If its showing advantageous outcomes then there is no need to further retest it and the tool is perfect for use.It should have some detail specification of the requirements and how its better in various application.

## VII REFERENCES

[1]. Harpreet kaur et al Int. Journal of Engineering Research and Application .http://www.ijera.com/papers/Vol3_issue5

[2]. Cucumber en.wikipedia.org/wiki/ Cucumber_(software).

[3]. Softwaretestinghelp.com/ selenium-webdriver-cucumber-selenium-tutorial-31

[4]. Selenium-toolsqa.com/cucumber/cucumber-tutorial toolsqa.com/selenium-webdriver/selenium-tutorial/

[5] . Testing tools-softwaretestinghelp.com/category/ software-testing-tools/

[6]. Automation testing-www.guru99.com/ automationtesting. Html

[7]. Lessons Learned in Software Testing, by C. Kaner, J. Bach, and B. Pettichord

[8]. Testing Computer Software, by C. Kaner, J. Falk, and H. Nguyen

[9]. Effective Software Testing, by E. Dustin.

[10].www.findwhitepapers.com/technology/software_development/software_testing.

[11]www.engpaper.com/free-research-papers-computer-science- software-testing.htm

[12]http://www.people.engr.ncsu.edu/txie/testingresearchsurvey.htm

[13].stackoverflow.com/ software testing.

[14] J. C. Huang, “An Approach to Program Testing,” ACM Computing Surveys, September 1975, pp.113

[15] E. F. Miller, “Introduction to Software Testing Technology,” Tutorial: Software Testing & Validation Testing.

[16] L. J. White and E. I. Cohen, “A Domain Strategy for Computer Program Testing,” IEEE Transactions on Software Engineering, May 1980, pp. 247-257.

[18]. J. A. Whittaker, “What is Software Testing? And Why Is It So Hard?” IEEE Software, January 2000.

[19].B. Beizer, “Software Testing Techniques,” Second Edition, Van Nostrand Reinhold Company Limited, 1990, ISBN 0-442-20672-0

[20]. D. Gelperin and B. Hetzel, “The Growth of Software Testing”, Communications of the ACM, Volume 31 Issue 6, June 1988, pp. 687-695

[21].N. Alshahwan and M. Harman. Automated web application testing using search based software engineering. In Proceedings of Automated Software Engineering, pages 3–12, 2011.

[22]. D. Amalfitano, A. R. Fasolino, and P. Tramontana. A

ISSN 2348 – 7968

GUI crawling-based technique for Android mobile application testing. In Proceedings of the IEEE International Conference on Software Testing, Verification and Validation Workshops, pages 252–261, 2011.

[23] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold, and P. Mcminn..survey of methodologies for automated software test case generation. Journal of Systems and Software, 86(8):1978–2001, Aug. 2013

. [24] S. Anand, M. Naik, M. J. Harrold, and H. Yang. Automated concolic testing of smartphone apps. In Proceedings of the ACM SIGSOFT Symposium on Foundations of Software Engineering, pages 59:1–59:11, 2012.

[25] J. H. Andrews, L. C. Briand, and Y. Labiche. Is mutation an appropriate tool for testing experiments? In Proceedings of the International Conference on Software Engineering, pages 402–411, May 2005.