ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 5, Issue 5, May 2016

Automated Model Based Testing for an Web Applications

Agasarpa Mounica, Lokanadham Naidu Vadlamudi

***Abstract*- As the development of web applications plays a major role in our day-to-day life. Modeling the applications should be appropriate, if not it leads the application for poor performance. These web applications became popular due to usage of user needs. Testing is very important to improve the quality of web applications in developing. In this paper, we propose automated model based testing and test case generation for the web applications to test. Model based testing can also be used with agile testing confirmation projects. Finite State Machine and State chart diagrams are used to navigate in between the web applications and also to know the behavior of it. Then using some automated model based testing tools, test cases are automatically generated in model and then evaluated the web application.**

***Index Terms*- Agile testing, Model based Testing, Web applications.**

I. INTRODUCTION

A. Agile Testing

Agile is a continuous and iterative approach to software development, that dependent on robust interaction and automation to keep on rating with dynamic environments. Agile development [6] approaches are successfully used to create and maintain software easily. Later on, the developed software is tested by some automation tools. Agile is for day-to-day development or testing activities. It detects problems and continuously improves with Sprints (run at full speed over a short distance). It mainly focuses on delivering the software with in time to make customer satisfied. Some agile testing strategies can also be used like Regression testing, Scenario based testing, Test Driven Development, Exploratory Testing, User Acceptance Testing and also we can assume a good toolset to support for collaboration and automation.

Manuscript received May, 2016

***Agasarpa Mounica, PG** Student, Department of Information Technology, Sree Vidyanikethan Engineering College (Autonomous), India.*

***Lokanadham Naidu Vadlamudi, Assistant Professor (SL)**, Department of Information Technology, Sree Vidyanikethan Engineering College (Autonomous), Tirupati, India.*

B. Model-based testing (MBT)

MBT is software testing in which test cases are generated or a part from model which describes some aspects of the System Under Test (SUT). Model-based testing is an application for designing and executing objects to perform testing (includes Test cases to perform each and every object). Models are used to know the performance of SUT or to know the testing strategies with test environment. It is a kind of testing where the whole task or some tasks of the test cases [1] are getting generated which describes the performance of the system under test.

C. MBT in Agile Projects

MBT is very suitable for the development where agile methodology is being followed. In Agile methodology changes are highly accepted by the customer need. The independent testing team never can come up with the latest changes made. Moreover, developers' try to create the necessary documents at that point of time and it affects the test report for the testing team at large area. Here MBT comes up to reduce the separation problem. In agile [5] after each iteration, the application should be tested and it should not the demand for much effort to equal with the agile methodology. MBT is a system developing task for testing not for the test case generation as it is usually followed. So it is perfect to use MBT in agile projects where tasks can be divided into sub tasks. Testing development using MBT techniques to estimate the effort and also to monitor agile process.MBT is of two types as shown in fig.1

i. Online MBT

Test case generation [1] and execution in action, next is design after the output is received. Testing is done for infinite test suites.

ii. Offline MBT

Generating finite number of tests and executing them. Automate test case generation also allows execution in third party platform. It takes a tool for test execution.

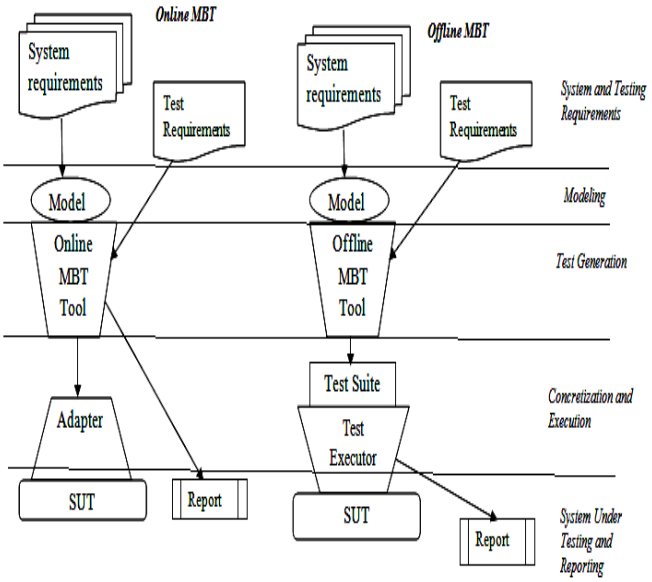

Fig.1 Types of Model based Testing

II. MODEL BASED TESTING PROCESS

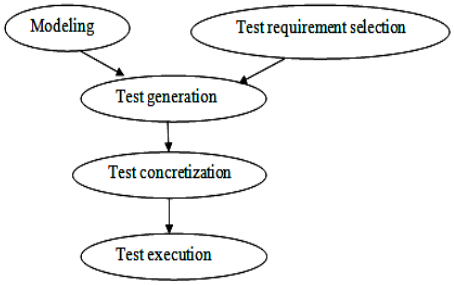

Fig.2 MBT Process

From the above fig.2 **[**2]

1. Modeling:

Purpose:

- Describes the system constraints for test generator
- Many faults can be found in this phase

Important:

- High level of abstraction
- The model also includes expected output of the SUT
- Two main aspects

1. Design model
2. Test model

2. Test requirements specification

Purpose:

- Guides the test generation

Three main categories of requirements specification are:
1. Aim for the model
2. Coverage criterion
   - State coverage
   - Transition coverage
   - Code coverage
3. Walking algorithms
   - Random walking
   - Coverage guided

3. Test generation

Purpose:

- Design and test

Offline

- Searching algorithms
- Tests are written in resolute format

Online

- Walking algorithms or searching algorithms
- The next step will be decided after the execution and output value received from the previous one.
- Algorithms should be fast

4. Test concretization

Purpose:

- Concretize test suite to executable level

Tools are providing

- Different test exporting formats
- Do-it-yourself by adding plug-ins if necessary.

5. Test execution

Purpose:

- Test executing and comparing the actual output to the expected output

Offline

- Tests will run in peripheral test execution platform
- In the platform the document and analyzed results can be known.

Online

- Tests are executed during the Test design
- MBT tools will write down a report and then analyze the results
- It makes possible to handle SUT

Analyzing results

- Traceability
- Reporting

III. AUTOMATED MODEL BASED TESTING

Automated Testing or Test Automation means testing any software product or application with some software tools or codes which requires no input manually. There are two general approaches for test automation:

1. Code-Driven Testing: Tested with different input arguments to authenticate the results that are correct or not.
2. Graphical User Interface Testing: Detects the changes and validate the performance of program is correct or not.

IV. FINITE STATE MACHINES (FSM'S)

Finite State Machines [3] is used to model certain application dependencies. It consists of set of states, inputs and outputs. By using set of inputs, it reasons to transition from one state to another and also gives a set of outputs. FSM models are extensively used to test a software application whether an implementation of conforms to specifications. It is used to explain the requirements. In addition to it, the transitions in FSM will encapsulate the desired performance for an implementation. Decisions regarding the rightness of the implementation are known by comparison of the expected behavior with the actual behavior by this FSM. It is based upon the approaches of the performance testing for Web applications. These approaches make use of probabilistic FSM called a **Markov chain**.

Markov chain means that information of preceding state is unrelated in expecting the probability of following states. So that, next state should depend completely on the current state. A Markov Chain can be analyzed statistically but not be predicted precisely, defined on a set of states with matrix of transition probabilities. A one dimensional random walk can also be appeared at Markov chain whose state space is given as a=0, +1, +2, +3, +4, ….. b=0.+1,+2,+3,…..For some number Probability "p" should satisfy 0<p<1, the transition probabilities (the probability $P_{a,b}$ of moving from state a to state b) are given by

$$P_{a,b+1} = P$$
$$P = 1 - P_{a,b-1}$$

According to Markov Chain the web application [4] is explained by states and transitions. "States' are used to signify the different application types that which are supported by the Web and "Transitions" are used to signify the navigation in between the state models that is from one application to another within it. Similarly in non-probabilistic of FSM, "Transitions" will correspond for the probability of "sum of states" that which are outgoing. The transition probabilities will give the correspondence user transition of a state from all. By traversing these Markov chain, traceability can be created based on the performance testing. In FSM, transition refers to the change of state caused by actions and it represents a condition to enable the transition. A transition function is the process, in which the current state is an input action and it returns the next state with new output actions. It also is used for mapping by arranging them in an sequence of input actions and output actions. Action means the movement that is to be performed at a certain moment. There are three types of actions:

1. Entry Action
2. Input Action
3. Output Action

Entry action:	Depends on present state
Input action:	Depends on input conditions and present state
Output action:	Performs when exiting the present state

"State transition diagram and table" are used to define the FSM. In these state transition diagram,

States	It is Represented in nodes that is number or words
Transitions	It is Represented in links that join the states
Input and Output Actions	It is Represented in letters or words on the arc of the transition which is separated by a slash ("input/output").

A. Finite State Machine Model Based Testing

A FSM model is generated as graph or state chart with a limited number of nodes that is States are interconnected by means of directed edges i.e., Transitions –> actions. When considering a web application we can easily propose a model of the application in terms of FSM's by following a few simple rules:

1 Each page of the application can be equated with a State of the application (in a black box sense).

2 Each tab of each page can be considered as a sub-State of that page.

3 Every action that alters or changes the page of the application in a way the results in a State changes.

4 Every action taken can be equated with a Transition for the purposes of the model.

As there are many models in web related applications. It can be used to find unknown relationships between pages and to identify important position of the web. Consider set of web pages as shown in fig.3 and construct the hyperlink graph. A node will be created for every web page [6] and also a directed edge. These directed edges are placed in between the two nodes if hyper-link of corresponding Web Pages is there. Assume a small document or model consisting of six web pages linked.

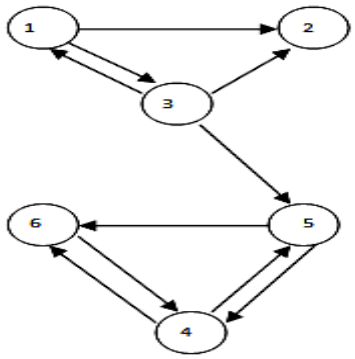

Fig.3 State Machine of six web pages

$$P = \begin{pmatrix} 0 & ½ & ½ & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & ½ & ½ \\ 0 & 0 & 0 & ½ & 0 & ½ \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Fig.4 Transition Matrix of web pages

From the fig.4. Square matrix P is the probability whose element is "$P_{a,b}$" moving from one state a (page a) to another state b (page b). For instance, let us assume that starting from one node in webpage to reach at another node is equally done by following the "outgoing links" from the node. Some rows of the matrix, such as row 2 in above example contain all zeros. This occurs whenever a node does not contain out links, many of such nodes are existing on the web. Such nodes are known as dangling nodes. One solution is to replace all zero rows as 0^T with ¼ e^T , where these e^T represents row vector of all ones and n represents the order of the matrix.

B. Requirements

The requirements to perform the model based testing in your system are:

Hardware configuration
- Processor : Intel i3
- RAM :4GB
- Hard Disk : 500 GB

Software configuration
- Operating System : 32 bit Windows7
- Tool : Selenium
- Platform : Java

C. Model based Testing Tool

Selenium

Selenium is an open source tool and can run on multiple browsers which supports for Cross Browser Testing. Customers will guide for developing an automation test for web applications. It also allows us in writing scripting for different languages like Java, C#, Ruby PHP and Python. The command sets are called **Selenese,** these create a language for testing. Selenium as four types of components: Selenium IDE, Selenium Remote Control, Selenium Web Driver [8] and Selenium Grid. This tool provides all types of Web applications with a variety of functions for testing scenarios.

Selenium IDE	**Records** the user performance and **play back** them.
Selenium Remote Control	Easy to convert Selenese commands into a programming language
Selenium Web driver	Accepts the commands and send it to the browser
Selenium Grid	Run tests on multiple machines in parallel

Sample program for Selenium tool in testing an application as shown in fig.5

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;


public class webdriver {


    public static void main(String[] args)
    {

        System.setProperty("webdriver.chrome.driver", "E:/chromedriver.exe");
        WebDriver driver= new ChromeDriver();

        driver.get("http://www.google.co.in/");
        driver.close();
        System.out.println("Successfully Selenium Webdriver is running");
    }

}
```

Fig.5 Program for Selenium in Google chrome

From the above fig. it is clear that by using selenium tool we can test any type of web applications easily. Hence, we can say that automated tool of selenium is helpful in writing the code for web applications and also in testing it.

V. CONCLUSION

A Model for web application will be obtained first. Later on, test cases are performed for testing an application by automated model based testing tool. Selenium is the best model based testing tool as well as free source to test. Using these selenium tool the model of web application is executed in test scripts.

ACKNOWLEDGEMENT

We would like to express our thankful researcher's, those who provided significant help. Mr. K.K. Basheer Assistant Professor, we are extremely thankful for your support and suggestions and also Dr. K. Ramani Professor, we are fully indebted of your advice and encouragement

REFERENCES

[1] Vikas Suhag and Rajesh Bhatia," Model based Test cases Generation for Web Applications", International Journal Of Computer Applications, Vol.92, No.3, April 2014.
[2] Sumit Machra and Narendra Khatri," Model Based Testing Of Website", International Journal on

Computational Sciences & Applications, Vol.4, No.1, February 2014.

[3] Kulvinder Singh and Semmi," Finite State Machine based Testing of Web Applications", International Journal of Software and Web Sciences, 2013.

[4] Tie Liu,"Application of Markov Chains to Analyze and Predict the time Series", in Modern Applied Science, Vol.4, No.5, May2010.

[5] Bernhard Rumpe,"Agile Test based Modeling", International Conference on Software Engineering Research and Practice, June 2006.

[6] David Talby and Arie Keren, Orit Hazzan and Yael Dubinsky," Agile Software Testing in a Large-Scale Project", IEEE Software, July 2006.

[7] Harry Robinson," Graph Theory Techniques in Model Based Testing", International Conference on Testing Computer Software, 1999.

[8] http://www.faichi.com/blog/automated-selenium-testing-framework-using-testng-webdriver