Volume 3, Issue 5, May 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Testing Web Applications Using UIO with GA

Poonam Soni*
M.Tech (CSE), Department of Computer Science and Applications, K.U., Kurukshetra, Haryana India.

Dr. Sanjay Tyagi
Assistant Professor, Department of Computer Science and Applications, K.U., Kurukshetra, Haryana India.

Abstract*— The Internet uses are increasing day by day. The World Wide Web has become global system for providing information and services. The technological evolution however is not supported by adequate web testing methodologies. Usually web testing is carried out without following any well-defined procedure, it lacks suitable tool support. Model Based Testing has gained attention with popularization of modelling in software development & model simplifies and optimizes testing. This paper describes how to model a web application into Finite State Model and generate Unique-Input-Output (UIO) for each state. After generating UIOs, test sequences are computed. In this paper, Best test sequences are showed on the basis of state covered.

Keywords*— Finite State Machine, Model Based Testing, Test Sequences, Unique Input Output, Web Application Testing.

I. INTRODUCTION

The Web has had a significant impact on all aspects of our society, from business, education, government, entertainment sectors, industry, to our personal lives. The main advantages of adopting the Web for developing software products include (1) no installation costs, (2) automatic upgrade with new features for all users, and (3) universal access from any machine. The difficulty in testing web applications [2] is many-fold. First, web applications are distributed through a client/ server architecture, with (asynchronous) HTTP request/response calls to synchronize the application state. Second, they are heterogeneous. During the past decade, researchers in increasing numbers, have proposed different techniques for analysing and testing these dynamic, fast evolving software systems. Web application testing is more complex than software testing because they have many different communication protocols, hardware platforms, operating systems and browsers. The growing complexities increase the testing time and cost associated with it. Hence, there is demand for effective testing approaches to manage the growing complex web application to assure quality of business. Model based testing[4] has proven the capability to provide remarkable improvements in minimizing cost, increased quality and reduced testing time.

Finite state machines (FSMs) have been used to model systems in different areas like sequential circuits, software development and communication protocols. Usually the implementation of a system specified by the FSM is tested for conformance by applying a sequence of inputs and verifying that the corresponding sequence of outputs is that which is expected. This paper focuses on the U-method for test sequence generation where unique input/output (UIO) [5] sequences for each state have to be generated. The problem of generating such sequences is known to be NP-hard. The primary contributions of this paper are showing how UIO generation can be formulated using an algorithm and that by using UIOs, test sequences are generated.

II. BACKGROUND

Testing is an important part of software engineering process and it account up to 50% of total cost of software development. This leads to study of Finite state Machine to ensure the correct functioning of the system. The generation of effective test sequences is very important for conformation testing. Test Sequences can be generated by many methods such as Transition Tour (T-method), Distinguish Sequences method, Unique Input Output method (UIO-method) [6], Characterizing set (W-method). In all these UIO method is very popular because: T-method does not consider transfer faults and W-method is effective in revealing operation & transfer errors, but number of tests generated by this method is usually large.

A. Finite State Machine

Many devices used in daily life contain embedded computers. An embedded computer often receives inputs and responses with appropriate actions. While doing so, it moves from one state to another. Behaviours of these systems are often modelled by FSM (Finite State Machine). A FSM [7] is represented by 6-tuples (X, Y, Q, q_0, δ, O), where X is a finite set of input symbols also known as input alphabet, Y is a finite set of output symbols also known as output alphabet, Q is finite set of states, $q_0 \in Q$ is initial state, δ: $Q \times X \rightarrow Q$ is the state transition function, O: $Q \times X \rightarrow Y$ is an output function. FSM can be represented using state transition diagram, where vertices correspond to state and edge to transition which are labelled with associated input output (Fig. 1).

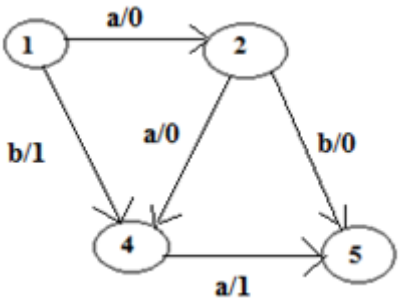

Fig.1 Deterministic FSM

An FSM is said to be deterministic, if there is no pair of transition that have same initial states and same inputs. FSMs for which a transition exists for every input a ϵ X and state s ϵ S are known as completely (fully) specified. An FSM M is strongly connected if for every pair of states (s_i, s_j) from M there is some input sequence that takes M from s_i to s_j. Fig. 1 represents deterministic FSM.

B. Conformance Testing

When testing from an FSM[3] model M, it is assumed that the implementation under test (IUT) can be modelled by an unknown FSM M1 and thus that testing involves comparing the behaviour of two FSMs. Verifying that M1 is equivalent to M by only observing the input/output behaviour of M1 is known as conformance testing or fault detection. Fault can be categorized as operation error /output error or transfer error [6]. Operation errors are those where wrong outputs are produced and transfer errors are error in the transition from one state to another. An output fault can be detected by executing a transition and observing its output. A state transfer fault can be detected by checking if the final state is correct after the transition testing is applied. There are three main techniques that can be used in state verification:

- Distinguishing sequence (DS)
- Unique input/output sequence (UIO)
- Characterizing set (CS)

A distinguishing sequence is an input sequence that produces output unique for each state. Not all FSMs have a DS. A UIO sequences is a sequence of input and output pairs that distinguishes a state of an FSM from the remaining states. A UIO sequence of length k for some state s ϵ Q, is denoted as UIO(s) and looks like the following sequence:

$$UIO(s) = i_1/o_1.i_2/o_2\ldots\ldots\ldots i_{(k-1)}/o_{(k-1)}.i_k/o_k.$$

UIO(s), is a sequence of one or more edge labels such that have following condition holds:

$$\delta(s, in(UIO(s)) \neq \delta(t, in(UIO(s)), \text{ for all } t \in Q, t \neq s \text{ [6]}$$

A characterizing set is a set of input sequences W which can distinguish any pair of states. If every sequence in W is executed from some state s_j, the set of output sequences verifies s_j. However, this technique requires a number of input sequences to be executed for each state, and therefore could lead to long test sequences.

Fig. 2 Presents the Gmail System

III. MODELLING WEB APPLICATION

A Web Application involves organizing information and navigation mechanisms so that user may find and use that information effectively. The purpose of this workflow detail is to organize the Website's content and features into a logical structure of the whole Website. For testing Web-based application, Gmail System is considered which helps to user for mailing to one another. Users first go to Gmail page then login through new or already registered transition.
Firstly, here one module is taken this is represented by FSM with input-output pairs. Test Sequences are generated using UIO (Unique-Input-Output method) for this module. The Gmail System is shown in above Fig. 2.

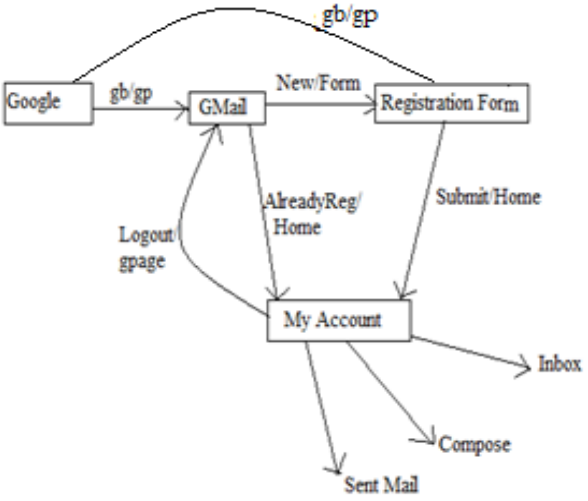

Fig. 3 FSM of Gmail System

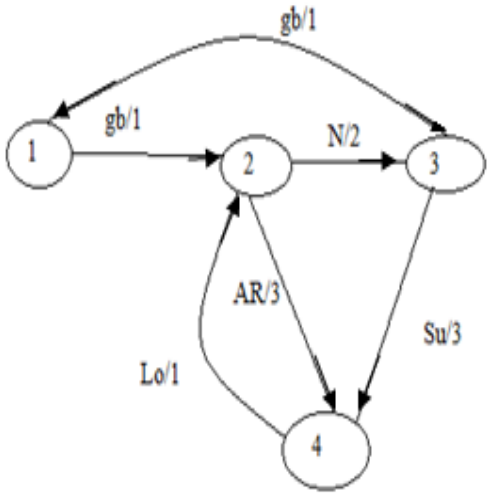

Fig. 4 FSM of above System

In above Fig. 4 gb, N, AR, Su & Lo represents gbutton, New, Already Registered, Submit and Logout input alphabets respectively. 1, 2, 3 are the output alphabets representing Gpage, Form and Home respectively.

Node 1--- Google
Node 2--- Gmail
Node 3--- Registration Form
Node 4--- My Account

IV. TEST GENERATION

Let E denotes the set of core edges in system module. Recall that edges corresponding to a reset input in each state are included in E

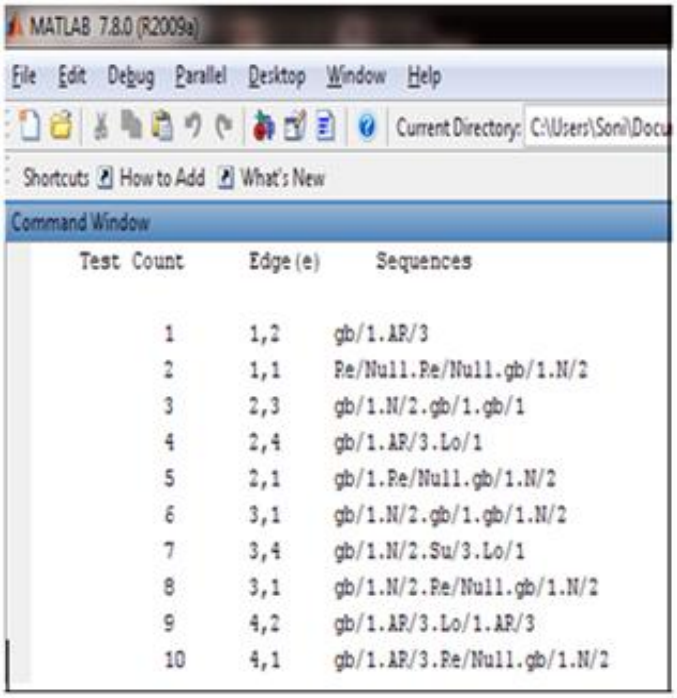

Fig. 5 Test Sequences produced

No. of Test Sequences = No. of core edges + No. of states (edge corresponding to reset input for each state)
The following procedure [6] is used to construct tour of an edge:

1. Find the UIO for each state in M.
2. Find the shortest path from the initial state to each of the remaining state. Shortest path from initial state q_1 to any other state $q_i \in Q$ is denoted by $P_i(q_i)$.
3. Construct an edge tour for each edge in Machine. Let TE (e) denotes a subsequence that generates a tour for edge e.
   $TE(e) = P_{head(e)}(1).label(e).UIO(tail(e))$

Test Sequences that are generated by above algorithm have been implemented in MATLAB as in Fig. 5.

V. TESTING WEB APPLICATION USING GA

Generation of test cases uses both the test model and the test criteria [8]. Generation proceeds in two steps:

1. Path Selection: Path selection is based on the required coverage criterion of each FSM. This is easily automatable via existing Unique Input/output (UIO) algorithms. Path selection algorithms that do not end in the last state require the addition of a proper sequence to an end state. Otherwise, the sequence cannot be combined properly with the follow-on test sequence in the aggregate test sequence.
2. Test case selection: The test cases are generated by the UIO technique of the FSM and Input constraints decide which inputs to select and in which order. One issue is to satisfy all constraints and criteria during selection.
3. Fitness function: Fitness function is calculated from the transition ranking process. The transition ranking process ranks each input/output pair of the FSM according to number of times it reoccurs in the transition table of the FSM. If any input/output pair occurs only once that pair gets the lowest rank, a pair that occurs twice is ranked next etc. If some pairs have the same number of occurrences in the transition table, they get the same rank [1]. Fig. 6 shows the sequences and fitness.

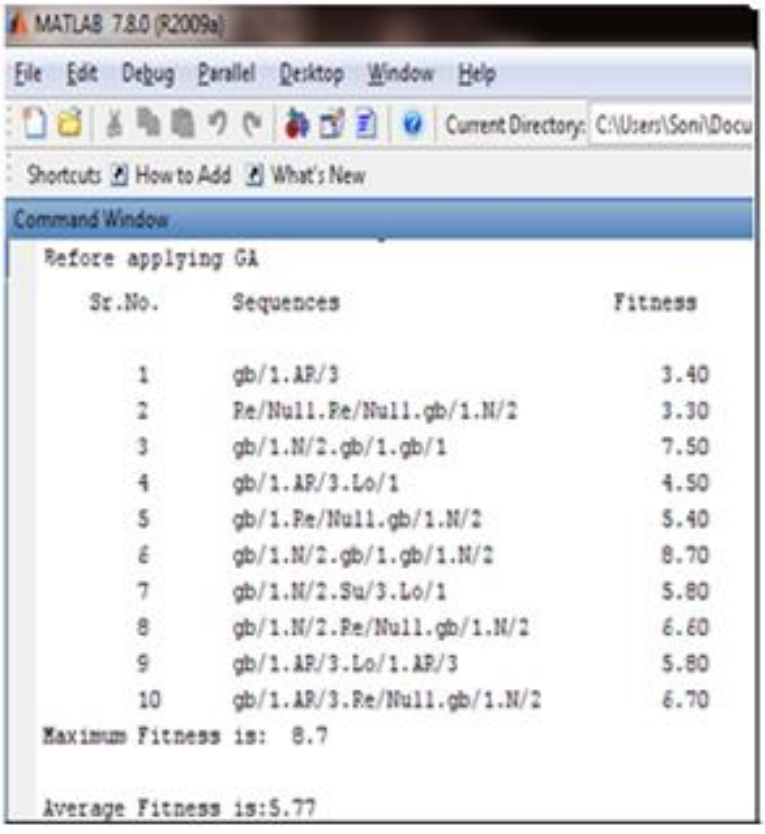

Fig. 6 Before applying GA

Google Page acts as initial state of the FSM. The hyperlinks and the hypertext act as the edges of the FSM and these edges connect the two web pages which are states in the FSM.
The Proposed framework for automatic test case generation using GA and UIO of FSM is:

1. Generate FSM of a module of Web Application.
2. Generate Test Cases using UIO for FSM
3. While(Not(terminating condition))d{
4. Generate Next generation using selection criteria as described above (Fitness Function).}

After these steps we get best test sequences which are below are shown in Fig. 7, Fig. 8 & Fig. 9. These sequences have higher chances of getting selected.

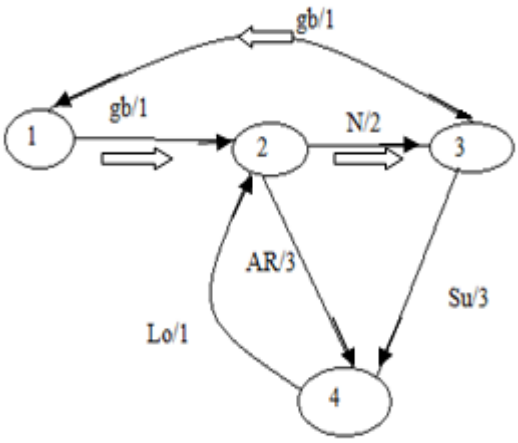

Fig. 7 Test Sequence: gb/1.N/2.gb/1.gb/1.N/2 (Fitness 8.7)

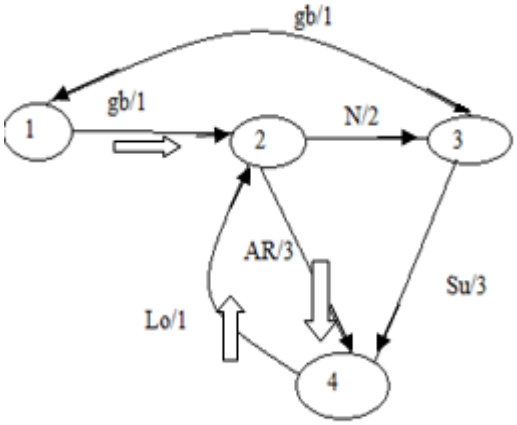

Fig. 8 Test Sequence: gb/1.AR/3.Lo/1.AR/3.Re/Null (Fitness 5.8)

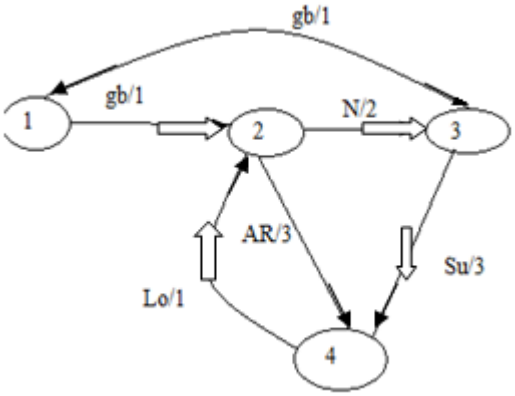

Fig. 9 Test Sequence: gb/1.N/2.Su/3.Lo/1.Re/Null (Fitness 5.8)

VI. CONCLUSIONS

This paper describes how web application is converted into Finite State Machines without using source code. It also describes the procedure for generating test sequences using UIO (Unique Input Output). It then shows the best test sequences that are selected on the basis of fitness that are calculated no. of times that input-output pairs occurs in transition table. Here only selection criterion is used in GA. The data is chosen from a collection of possible values gives by the tester, and the constraints are used to ensure that the data is valid for the test requirement.

REFERENCES

[1] Anneliese A. Andrews, Jeff Offutt, Roger T. Alexander, "Testing Web applications by Modeling with FSMs", In: Software Systems and Modeling. Vol. 4, No. 3, Pages 326-345, July 2005.

[2] Filippo Ricca, Paolo Tonella, "Analysis and Testing of Web Applications", In: Proceeding of the 23rd International Conference on Software Engineering ICSE '01, IEEE Computer society, July 2001.

[3] Huaikou Miao, Zhongsheng qian, Tao He, "Modeling Web Browser Interactions using FSM", In: Proceeding of IEEE Asia-Pacific services Computing Conference, 2007.

[4] Ibrahim K. El-Far and James A. Whittaker, "Model Based Software Testing", Wiley, 2001.

[5] Karnig Derderian, Robert M. Hierons, Mark Harman and Qiang Guo, "Automated Unique Input Output sequence generation for conformance testing of FSMs", The Computer Journal, Vol. 00, No. 0, 2006.

[6] Mathur, A. P., "Foundation of Software Testing", Pearson Education Publication, 2008.

[7] Quian Zhongshen, "An Approach to Testing Web Applications Based on Probable FSM", IEEE Computer Society, 2009.

[8] Singh, K. and Kumar, R. "Design of Fault Tolerance System Using Genetic Algorithm Employing Mutation and Back-to-Back Testing" International Journal of Computer Theory and Engineering (IJCTE), 1793-8201 (Print Version), 2009.

