Finite State Machine Based Modeling and Testing of Web Based Applications using Mutation Analysis

Nipur[a], Rakesh Kumar [b] and Priyanka Gupta[c]
[a] *Department of Computer Science and Applications, Kanya Gurukul Mahavidyalaya, Dehradoon, Uttaranchal, India.*
[b]*Department of Computer Science and Applications, Kurukshetra University, Kurukshetra, Haryana, India.*
[c]*Department of Computer Applications, Maharaja Agrasen Institute of Management and Technology, Old Saharanpur Road, Agrasen Chowk, Jagadhri, Haryana, India.*
*gupta800@gmail.com

Abstract

Increasing popularity of web applications demands for their better testing that can lead to uninterrupted usage. This paper suggests the testing of web applications by using black box testing mechanism. A testing methodology that focuses on finding errors of omission for web applications has been suggested using specifications and mutation analysis. Functional behavior of the application is modeled in the form of a Finite State Machine. Mutation operators based on specifications are used to generate mutated versions of model FSM. W method of test case generation is used for generating test cases. Model FSM and mutated versions are excited using generated test cases and their comparative analysis is used to locate errors in sample web application "Online Journal".

Keywords - Finite state machine based testing, Mutation specification testing, Web applications testing.

Introduction

Testing plays an important role for assuring quality of any software. Success of testing depends upon identifying quality test cases. There are a number of testing techniques available that generates test cases and focuses on detecting errors of some particular category. Basically errors could either be errors of omission or errors of commission. Taken as a universe of program behaviors if S is the expected behavior known as specification and P is the observed behavior called implementation and T is the set of test cases; the relationship between all could be well analyzed from the Figure 1. There are certain specified behaviors but are not implemented (5,4) represents errors of omissions; certain programmed behaviors that were not specified (6,3) represents errors of commission (Jorgensen, 2008).

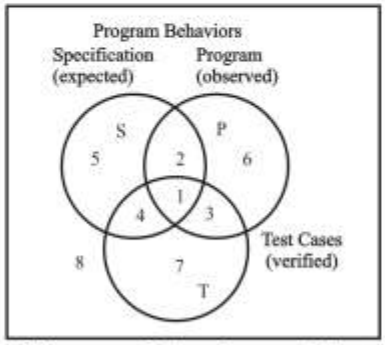

Figure 1: Relationship of Program Behaviors and Test cases (Jorgensen, 2008).

Test cases in T are identified by a testing method. Test cases for (5,4) could only come from black box testing and for (6, 3) only from white box testing mechanism. Black box testing mechanisms works on specifications rather than implementations. Specification based testing techniques faces the problem of redundancy and gaps in test cases identification. Because of this to complete the set of test cases through such techniques becomes a challenge. Mutation analysis is a test case adequacy criteria generally used with white box testing techniques to ensure the completeness and effectiveness of test cases.

Demillo *et al*., (1978) And Budd *et al*., (1980) have discussed Mutation as a fault-based testing technique. According to Morell (1990) fault-based testing helps to find out the absence of pre specified faults in a program. Mutation-based testing works with a set of operators. Each of the operators modifies the source code as if an error has been injected. The modified program is known as a mutant. Test cases are generated and both mutants and original program are executed against them. If any test case can produce different results then mutant is said to be killed. Otherwise, the mutant is live. Either the mutant program is found equivalent to the original program or the test data set is not adequate and needs enhancement. The adequacy of a test data set is measured by a mutation score (MS), which is the ratio of the number of killed mutants to the total number of non-equivalent mutants.

Mutation Testing is based on two basic Assumptions:

(a) *The Competent Programmer Hypothesis:* In general programmers are competent. i.e., the programs they write are nearly correct. And the program which may be corrupted has only very small mistakes (Demilo, 1978; Patrick, 1997).

(b) *The Coupling Effect Hypothesis:* Large program faults, which are semantic in nature depends upon smaller syntactic faults and can be detected with mutation testing. (Patrick, *et al*., 1997; How Tai Wah, 2003).

This work targets on technique to identify test cases for detecting errors of omission. Such errors are common in all type of applications whether it is desktop, web, network or any other. Web applications and their use are growing tremendously all around and because of technological advancements and cheaper access they are now becoming larger, interactive, and essential to the international use of computers. A lot of business loss could happen if a web based application could not serve the purpose. Therefore it is worth to work on testing of web applications. A sample web application "online Journal" is used to explain the methodology.

In order to identify functional test cases for web application its mandatory to represent its specifications/functional areas in some formal way that could ease the job of test case identification. Specifications can be done by making use of many representations like event sequence graphs, finite state automata machines, Estelle, Petri nets, state charts, Specification and Description Language, Object Z specification, refinement calculus and many more. Author uses Finite State Machine to model the behavior of web application.

Therefore this paper works on testing of web based applications targeting to find some probable errors of omission using black box testing mechanism and in order to ensure the completeness of identified test cases mutation analysis for specifications is used on finite state machine based model of web application.

Mutation testing and specifications

Mutation testing is basically a white box method but it can be used as a black box testing strategy to check the effectiveness and completeness of test cases derived from specifications. Some of the researchers have worked in the testing field for specifications and have suggested some mutation operators for some specification languages.

Pinto Ferraz Fabbri *et al*., (1994) have discussed about the sequencing errors in FSM. They said operation errors, transfer errors and extra/missing states errors are commonly found in FSMs. They suggested some mutation operators for FSMs: arc missing, wrong starting state, event missing, event exchanged, event extra, state extra, output exchanged, output missing, output extra.

Black *et al*., (2000) have discussed the use of mutation analysis with model checkers which automatically generates complete test sets from formal specifications. They have suggested some mutation operators like ORO (operator replacement operator), SNO (simple expression negation operator), MCO (missing condition operator) for the same combination and have undergone theoretical as well as empirical comparison of suggested operators.

Abdurazik *et al*., (2000) have compared three specification based criteria named specification-mutation coverage, full predicate coverage, and transition-pair coverage. They have used Mathur and Wong's PROBSUBSUMES measure for the comparison. They found out that specification mutation PROBSUBSUMES full predicate and full predicate PROBSUBSUMES specification mutation, however, neither the full predicate tests nor the specification mutation tests had high transition-pair scores, and the transition-pair tests did not have high full predicate or specification mutation scores. This shows that transition-pair tests offer something different from full predicate and specification mutation tests and the later two are almost similar.

General mutation operators for event sequence graph as suggested by Belli *et al*., (2006) are: arc insertion, arc omission, event insertion, event omission. They applied the operations on ESG, FSA, State charts. They concluded that mutants based on ESG and mutants based on state charts do not differ much in their fault detection capability.

Beyazıt *et al*., (2010) have suggested three mutation operators for ESGs (1) Insert arc (iA) operator adds a new non-pseudo arc to the model, (2) delete arc (dA) operator removes an existing arc from the model, and (3) reverse arc (rA) operator reverses the direction of an existing non-pseudo arc in the model. Also they have suggested three mutation operators for MFSMs (1)Insert transition (iT) operator adds a new transition to the model, (2) delete transition(dT) operator deletes a transition from the model, and (3) reverse transition (rT).Operator reverses an existing transition in the model. They have given a formula to put down an upper limit to the number of first order mutation operators.

Finite state machine and its mutations

A finite state machine is a Six-tuple (X1, X2, Q, qo, S, O), where X1: is a finite set of input symbols also known as the input alphabet .X2: is a finite set of output symbols also known as the Output alphabet. Q: is a finite set of states. qo ϵ Q: is the initial state. S: $Q \times X1 \rightarrow Q$ is a next state or state transition function. O: $Q \times X1 \rightarrow X2$ is an output function (Mathur, 2008).

A FSM is said to be completely specified if from each state in FSM there exists a transition for each input symbol. A FSM is strongly connected if every state is reachable from the initial state. In a FSM two states are said to be V-equivalence if two states excited yield identical output sequences. Two states in FSM are said to be K-equivalence if when excited by any input of length k yields identical output sequences. A FSM is considered minimal if the number of states in it is less than or equal to any other FSM equivalent to it.

A FSM represents the design of the implementation (final system). In order to confirm that the implementation confirms to the specifications, in spite of using implementation, FSM could be used. Possible errors that could be there in system could easily be represented by mutated FSM's (Mathur, 2008). Some probable mutations that can be made to the FSM are insert extra state, delete existing state,

insert extra transition, delete existing transition etc. In figure FSM represents the actual model and its mutation M' represents operation error and extra state error in the implementation.

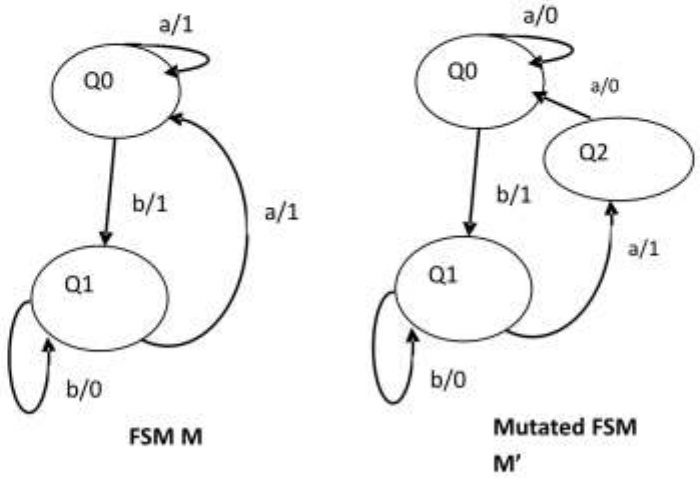

Figure 2: FSM and its Mutation.

Representing Web Application as an FSM

Any web application is an active sequence of user interactions with the application through some interface provided by the application. These external events change the states of the WAs. Such changes in states of WAs can easily be modeled with the help of Finite State Machines.

Even the smallest possible real life web application could have large number of states to be represented in FSM; the problem of state explosion is obvious. Therefore hierarchical representation of FSMs are recommended for modeling web applications where lower level FSMs represent the complete behaviors for the subparts of the web application and could be represented as a single state at higher level FSM for the upper level part of the web application. This gives a neat and understandable representation of web application into FSM. This way the behavior/ services provided by the WA are represented as states in FSM and the movement between different pages of WA can be represented by transitions between states in FSM.

This work discusses the modeling and testing of "online journal publication" a student's project for better understanding of the introduced concepts. This web application provides services in the form of three major types of users: author, reviewer and general user of the application. This application provides different services for different categories. Authors themselves need to be registered and can upload their paper for submission, view other papers; Reviewers are the special type of users who accesses the papers in different categories and reviews them and finally the papers are uploaded back for the access of all others. General users need to subscribe to access the material available with the application. Different categories of papers are placed under different headings like bioscience, computer science, Software Engineering, Databases, Nano Science etc. files can be uploaded in different formats pdf, txt, docx, doc, rtf. Services provided directly by the application taken as logical pages modeled as states in FSM are shown in table.

Table 1: Represents FSM of web application "Online Journal".

		Next state/output																	
Stat no.	**State name**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**PI**	**AN &PW**	**RN & PW**	**UN & PW**	**PD & id**	**RC**	**CR**	**DOC**	**SR &id**	**art**	**click**	**C7**
S1	Home	S2	S11	S12	S10/ PD	S13	S14	S3/ID	-	-	-	-	-	-	-	S9/SD	-	-	S15
S2	Login	-	S11	S12	S10/ PD		S14	-	S4	S5	S6	-	-	-	-	-	-	S1	S15
S3	Registration	S2	S11	S12	S10/ PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S4	Author	-	S11	S12	S10/ PD	S13	S14	-	-	-	-	-	-	S8 /RC	S7/ UC, art	S9/SD	S8/doc	S1	S15
S5	Reviewer	-	S11	S12	S10/ PD	S13	S14	-	-	-	-	-	S7 /UC, art	-	S7/ UC, art	S9/SD	S8/ doc	S1	S15
S6	General User	-	S11	S12	S10/ PD	S13	S14	-	-	-	-	-	-	-	-	S9/SD	S8/doc	S1	S15
S7	Upload	S2	S11	S12	-	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S8	Download	S2	S11	S12	-	S13	S14	-	-	-	-	-	-	--	-	-	-	S1	S15
S9	Subscribe	S2	S11	S12	S10/ PD	S13	S14	-	S4	S5	S6	S10/ rec	-	-	-	-	-	S1	S15
S10	Payment mode	S2	S11	S12	-	S13	S14	--	-	-	-	-	-	-	-	-	-	S1	S15
S11	Contact us	S2	-	S12	S10/ PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S12	About us	S2	S11	-	S10/ PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S13	FAQ	S2	S11	S12	S10/ PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S14	Editorial Board	S2	S11	S12	S10/ PD	S13	-	-	-	-	-	-	-	-	-	-	-	S1	S1 5
S15	Log out	S2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	S1	-

Testing Web Application using Finite State Machine Model

The web application could have probable errors in it that means its behavior doesn't conform to the requirements. There could be various types of possible variations. Simplest one is not responding to the service in the fashion supposed to (incorrect output); another could be link validation (both static and dynamic) that signifies that when requested for some service through some hypertext, there should be a proper linkage to that service, it means transition from current position/page to the next correct position/page; it may be possible to have a page in the application that is not linked with the application through any means that's again an error and needs to be tested; other hand it may be possible that a page is missing, this situation arises when a hypertext is present somewhere but clicking on that hypertext doesn't land on any existing page.

Also some probable errors of omission in web applications are:

1. Not existing output function that leads to absence of correct output
2. Forgotten transitions, means hyperlinks to be placed at different locations of the web pages.
3. Forgotten functionalities in web application(missing pages)

Some possible errors in the sample web application: user asking for particular research paper in Computer science category but returned with some another (incorrect output); user clicked on hypertext mentioned registration form but landed in reviewer's area in spite of registration page; user demanding for his account details but user accounts page is missing and clicking on hypertext makes no transition. Such deviations from original web application can be represented by mutated FSM. These mutations could represent the probable variations (errors) that a web application could have from the original requirements. Mutations in the FSM can be done by making use of mutation operators.

Each of the mutation operators is supposed to imply some of the probable error situations as signified above. Some possibilities are 1) Erroneous output: an operator to induce a situation that being at some stage, having some input it produces an output that varies from the actual one. 2) Improper transition: an operator to induce a situation that when being at some state, having an acceptable input may land to some invalid state. 3) Missing page: an operator to induce a situation that being at some stage, having some input, the transition lands the user no where, which means no actual transition takes place to the required page and the page is not available. 4) Unwanted /unused pages: there could be some means to find out some unwanted/unused pages that are actually never demanded.

To test a WA means to determine whether it conforms to the requirements set for it or not. For that purpose a test case set needs to be generated. One way of generating these test cases could be using the model FSM for the web application. Test cases for model FSM could be generated by using any one of automata theoretic test generation techniques like W method, partial W method or UIO- sequence method. Test generated using W or Wp methods guarantee the detection of all missing transitions, incorrect transitions, extra or missing states and errors in the output associated with a transition (Mathur, 2008).

All the FSM including model FSM and mutated FSMs are activated from their initial states with these test cases generated. The output sequences in each case are recorded and compared. If the test case generates same output in any pair it signifies that the two FSMs are equivalent which means that they possess the same output behavior for that test case and hence no identified errors in web application otherwise a

mismatch signifies some particular error in web application. Following the above said model it will be possible to test any web application by representing it as a FSM. The model will be able to find out some possible error types available in the application and definitely will be able to find out the effective and complete test case set for the procedure.

Algorithm representing the methodology used for testing:

1. Represent the sample web application functional behavior in the form of a FSM.
2. Generate test cases for FSM by making use of W- method of test case generation.

2 (a) Derive characterization set (w-set) from the description of FSM, which is the set of input sequences that distinguish the behavior of pair of states in FSM.

1) Construct the sequence of k-equivalence partitions
2) Traverse these k-equivalence partitions in reverse order to obtain the distinguishing sequences for each pair of states which forms the elements of w-set.

2 (b) Construct the testing tree for FSM and from that tree constructs the transition cover set P. exciting an FSM with elements of P ensures that all states are reached and all transitions have been traversed at least once.

2 (c) Construct the set Z by making use of input alphabet X and characterization set w:

$Z = X^{0}.W$

2 (d) design the required test case T=P.Z

3. Generate first order mutants of FSM by making use of mutation operators like insert arc, delete arc, reverse arc, insert node, delete node.
4. Activate the model FSM (M) and Mutated versions of FSM (M1, M2…..) with the test set T.
5. Results are recorded and compared. Mismatch in M and any of Mi's(where i=1,2,….) signals an error. These errors could be one or more out of the set S={ operation error, transfer error, extra state error, missing state error}

Implementing proposed testing model on online journal web application

Sample web application online journal is tested for errors of omission by making use of the above said methodology. The web application in the form of a FSM is represented as a state transition table as in table P1. In order to construct the test set the first step is to find out the characterization set that distinguishes the states of the web application.

k-equivalence partition tables are constructed by making use of table P1. While calculations three partitions were formed P1= { 1, 2, 3,4, 5 ,6, 7} P2= { 1, 2, 3,4, 5 ,6, 7,8,9,10,11,12,13}and P3={ 1, 2, 3,4, 5 ,6, 7,8,9,10,11,12,13} The corresponding tables derived for k=1,2,3 are shown in tableP1,P2,P3. By making use of these partition tables a table representing distinguishing sequences for all pair of states in the FSM representing web application is derived and can be found in table 5. From this table finally the w-set is found which is { PI, C4, AN & PW CR, CR, RC, SR & ID, PD & ID, C2 C4} and it represents the set of strings that distinguishes all states of the web application.

From the model FSM a testing tree for the web application is generated and from that transition cover set for the application is derived which is represented as P={null, C1, C2,C3,C4,C5,C6 ,C7, PI, SR & ID,C1click, C1UN&PW, C1C2, C1C3, C1C4, C1C6, C1AN&PW, C1RN&PW, C1UN&PWC2, C1UN&PWC3, C1UN&PWC4, C1UN&PWC5, C1UN&PWC6, C1UN&PWSR&ID, C1UN&PWArt, C1UN&PWClick, C1UN&PWArtC1, C1UN&PWArtC 2 , C1UN&PWArtC3 ,

C1UN&PWArtC5, C1UN&PWArtC6 , C1UN&PWArtclick, C1AN&PWC2, C1AN&PWC3, C1AN&PWC4, C1AN&PWC5, C1AN&PWC6, C1AN&PW CR, C1AN&PWDoc, C1AN&PWSR&ID, C1AN&PWArt, C1AN&PWDocC1, C1AN&PWDocC2 , C1AN&PWDocC3 , C1AN&PWDocC5, C1AN&PWDocC6, C1AN&PWDocclick, C1RN&PWC2, C1RN&PWC3, C1RN&PWC4, C1RN&PWC5, C1RN&PWC6, C1RN&PWRC, C1RN&PWDoc, C1RN&PWSR&ID, C1RN&PWArt, C1RN&PWclick, PIClick , PIC6, PIC1, PIC2, PIC3, PIC4, IC5, C2C1, C2C3, C2C4, C2C5, C2C6, C2click, C3C1, C3C2,C3C4,C3C5,C3C6,C3click, C4C1, C4C2, C4C3, C4C5, C4C6, C4click, C5C1, C5C2, C5C3, C5C4, C5C5, C5C6, C5click, C6C1 , C6C2, C6C3, C6C4, C6C5, C6click,SR&IDC1 , SR&IDC2, SR&IDC3, SR&IDC4, SR&IDC5, SR&IDC6, SR&ID AN&PW, SR&ID RN & PW, SR&ID UN & PW, SR&ID PD & ID, SR&ID click, C7C1,C7click}. Exciting the FSM representing web application with elements of P ensures that all states are reached and all transitions have been traversed at least once.

Since the number of states in FSM and final web application are supposed to be same therefore the set $Z=X^0.W$ = {null}. {PI, C4, AN & PW CR, CR, RC, SR & ID, PD & ID, C2 C4}
= {null, PI, C4, AN & PW CR, CR, RC, SR & ID, PD & ID, C2 C4}

The desired test set for the web application is given by:

T = P.Z
={null, C1, C2,C3,C4,C5,C6 ,C7, PI, SR & ID,C1click, C1UN&PW, C1C2, C1C3, C1C4, C1C6, C1AN&PW, C1RN&PW, C1UN&PWC2, C1UN&PWC3, C1UN&PWC4, C1UN&PWC5, C1UN&PWC6, C1UN&PWSR&ID, C1UN&PWArt, C1UN&PWClick, C1UN&PWArtC1 , C1UN&PWArtC 2 , C1UN&PWArtC3 , C1UN&PWArtC5 , C1UN&PWArtC6 , C1UN&PWArtclick, C1AN&PWC2, C1AN&PWC3, C1AN&PWC4, C1AN&PWC5, C1AN&PWC6, C1AN&PW CR, C1AN&PWDoc, C1AN&PWSR&ID, C1AN&PWArt, C1AN&PWDocC1, C1AN&PWDocC2 , C1AN&PWDocC3 , C1AN&PWDocC5, C1AN&PWDocC6, C1AN&PWDocclick, C1RN&PWC2, C1RN&PWC3, C1RN&PWC4, C1RN&PWC5, C1RN&PWC6, C1RN&PWRC, C1RN&PWDoc, C1RN&PWSR&ID, C1RN&PWArt, C1RN&PWclick, PI Click , PIC6, PIC1, PIC2, PIC3, PIC4, C5,C2C1,C2C3,C2C4,C2C5,C2C6,C2click,C3C1,C3C2,C3C4,C3C5,C3C6,C3click, C4C1, C4C2, C4C3, C4C5, C4C6, C4click, C5C1, C5C2, C5C3, C5C4, C5C5, C5C6, C5click, C6C1 , C6C2, C6C3, C6C4, C6C5, C6click,SR&IDC1 , SR&IDC2, SR&IDC3, SR&IDC4, SR&IDC5, SR&IDC6, SR&ID AN&PW, SR&ID RN & PW, SR&ID UN & PW, SR&ID PD & ID, SR&ID click, C7C1,C7click} . {null, PI, C4, AN & PW CR, CR, RC, SR & ID, PD & ID, C2 C4}.

Table 2: P1.

		Next state/output																	
State no.	State name	C1	C2	C3	C4	C5	C6	PI	AN&PW	RN&PW	UN&PW	PD&id	RC	CR	DOC	SR&id	art	click	C7
S1	Home	S2	S1	S12	S10/PD	S13	S14	S3/ID	-	-	-	-	-	-	-	S9/SD	-	-	S15
S2	Login	-	S1	S12	S10/PD		S14	-	S4	S5	S6	-	-	-	-	-	-	S1	S15
S3	Registration	S2	S1	S12	S10/PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S4	Author	-	S1	S12	S10/PD	S13	S14	-	-	-	-	-	-	S8/RC	S7/UC,art	S9/SD	S8/doc	S1	S15
S5	Reviewer	-	S1	S12	S10/PD	S13	S14	-	-	-	-	-	S7/UC,art	-	S7/UC,art	S9/SD	S8/doc	S1	S15
S6	General User	-	S1	S12	S10/PD	S13	S14	-	-	-	-	-	-	-	-	S9/SD	S8/doc	S1	S15
S7	Upload	S2	S1	S12	-	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S8	Download	S2	S1	S12	-	S13	S14	-	-	-	-	-	-	--	-	-	-	S1	S15
S9	Subscribe	S2	S1	S12	S10/PD	S13	S14	-	S4	S5	S6	S10/rec	-	-	-	-	-	S1	S15
S10	Payment mode	S2	S1	S12	-	S13	S14	--	-	-	-	-	-	-	-	-	-	S1	S15
S11	Contact us	S2		S12	S10/PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S12	About us	S2	S1	-	S10/PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S13	FAQ	S2	S1	S12	S10/PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S14	Editorial Board	S2	S1	S12	S10/PD	S13	-	-	-	-	-	-	-	-	-	-	-	S1	S15
S15	Log out	S2		-	-	-	-	-	-	-	-	-	-	-	-	-	-	S1	-

Table 3: P2.

			Next state/output																
Grp no	State no.	State name	C1	C2	C3	C4	C5	C6	PI	AN&PW	RN&PW	UN&PW	PD&id	RC	CR	DOC	SR&id	art	click	C7
1	S1	Home	S2 2	S11 2	S12 2	S10/PD 7	S13 2	S14 2	S3/ID 2	-	-	-	-		-	-	S9/SD 6	-	-	S15 7
2	S2	Login	-	S11 2	S12 2	S10/PD 7		S14 2	-	S4 3	S5 4	S6 5	-		-	-	-	-	S1 1	S15 7
	S3	Registration	S2 2	S11 2	S12 2	S10/PD 7	S13 2	S14 2	-	-	-	-	-		-	-	-	-	S1 1	S15 7
3	S4	Author	-	S11 2	S12 2	S10/PD 7	S13 2	S14 2	-	-	-	-	-		S8/RC 7	S7/UC,art 7	S9/SD 6	S8/doc 7	S1 1	S15 7
4	S5	Reviewer	-	S11 2	S12 2	S10/PD 7	S13 2	S14 2	-	-	-	-	-	S7/UC,art	-	S7/UC,art 7	S9/SD 6	S8/ doc 7	S1 1	S15 7
5	S6	General User	-	S11 2	S12 2	S10/PD 7	S13 2	S14 2	-	-	-	-	-		-	-	S9/SD 6	S8/doc 7	S1 1	S15 7
7	S7	Upload	S2 2	S11 2	S12 2	-	S13 2	S14 2	-	-	-	-	-		-	-	-	-	S1 1	S15 7
	S8	Download	S2 2	S11 2	S12 2	-	S13 2	S14 2	-	-	-	-	-		--	-	-	-	S1 1	S15 7
6	S9	Subscribe	S2 2	S11 2	S12 2	S10/PD 7	S13 2	S14 2	-	S4 3	S5 4	S6 5	S10/rec 7		-	-	-	-	S1 1	S15 7
7	S10	Payment mode	S2 2	S11 2	S12 2	-	S13 2	S14 2	--	-	-	-	-		-	-	-	-	S1 1	S15 7
2	S11	Contact us	S2 2	-	S12 2	S10/PD 7	S13 2	S14 2	-	-	-	-	-		-	-	-	-	S1 1	S15 7
	S12	About us	S2 2	S11 2	-	S10/PD 7	S13 2	S14 2	-	-	-	-	-		-	-	-	-	S1 1	S15 7
	S13	FAQ	S2 2	S11 2	S12 2	S10/PD 7	S13 2	S14 2	-	-	-	-	-		-	-	-	-	S1 1	S15 7
	S14	Editorial Board	S2 2	S11 2	S12 2	S10/PD 7	S13 2	-	-	-	-	-	-		-	-	-	-	S1	S15
7	S15	Log out	S2 2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	S1 1	-

Table 4: P3

			Next state/output																
Grp no	State no.	State name	C1	C2	C3	C4	C5	C6	PI	AN&PW	RN&PW	UN&PW	PD&id	RC	CR	DOC	SR&id	art		C7
1	S1	Home	S2 2	S11 9	S12 10	S10/PD 7	S13 11	S14 12	S3/ID 3	-	-	-		-	-	-	S9/SD 8	-	-	S1 5 13
2	S2	Login	-	S11 9	S12 10	S10/PD 7		S14 12	-	S4 4	S5 5	S6 6		-	-	-	-	-	S1 1	S1 5 13
3	S3	Registration	S2 2	S11 9	S12 10	S10/PD 7	S13 11	S14 12	-	-	-	-		-	-	-	-	-	S1 1	S1 5 13
4	S4	Author	-	S11 9	S12 10	S10/PD 7	S13 11	S14 12	-	-	-	-		-	S8/RC 7	S7/UC,art 7	S9/SD 8	S8/doc 7	S1 1	S1 5 13
5	S5	Reviewer	-	S11 9	S12 10	S10/PD 7	S13 11	S14 12	-	-	-	-		S7/UC,art 7	-	S7/UC,art 7	S9/SD 8	S8/ doc 7	S1 1	S1 5 13
6	S6	General User	-	S11 9	S12 10	S10/PD 7	S13 11	S14 12	-	-	-	-		-	-	-	S9/SD 8	S8/doc 7	S1 1	S1 5 13
7	S7	Upload	S2 2	S11 9	S12 10	-	S13 11	S14 12	-	-	-	-		-	-	-	-	-	S1 1	S1 5 13
	S8	Download	S2 2	S11 9	S12 10	-	S13 11	S14 12	-	-	-	-		-	--	-	-	-	S1 1	S1 5 13
8	S9	Subscribe	S2 2	S11 9	S12 10	S10/PD 7	S13 11	S14 12	-	S4 4	S5 5	S6 6	S10/rec	-	-	-	-	-	S1 1	S1 5 13
7	S10	Payment mode	S2 2	S11 9	S12 10	-	S13 11	S14 12	--	-	-	-		-	-	-	-	-	S1 1	S1 5 13
9	S11	Contact us	S2 2	-	S12 10	S10/PD 7	S13 11	S14 12	-	-	-	-		-	-	-	-	-	S1 1	S1 5 13
10	S12	About us	S2 2	S11 9	-	S10/PD 7	S13 11	S14 12	-	-	-	-		-	-	-	-	-	S1 1	S1 5 13
11	S13	FAQ	S2 2	S11 9	S12 10	S10/PD 7	S13 11	S14 12	-	-	-	-		-	-	-	-	-	S1 1	S1 5 13
12	S14	Editorial Board	S2 2	S11 9	S12 10	S10/PD 7	S13 11	-	-	-	-	-		-	-	-	-	-	S1 1	S1 5 13
13	S15	Log out	S2 2	-	-	-	-	-	-	-	-	-		-	-	-	-	-	S1 1	S1 5 13

Table 5: Distinguishing sequences for all pair of states.

Si	Sj	x	O(Si,x)	O(Sj,x)
1	2	PI,SR & ID	ID,SD	Nil, Nil
1	3	PI,SR & ID	ID	Nil
1	4	PI, CR, Doc, Art	ID	Nil
1	5	PI, RC, Doc, Art	ID	Nil
1	6	PI, Art	ID	Nil
1	7	C4,PI, SR & ID	PD	Nil
1	8	C4,PI, SR & ID	PD	Nil
1	9	PI, PD & ID, SR & ID	ID	Nil
1	10	C4,PI, SR & ID	PD	Nil
1	11	PI, SR & ID	ID	Nil
1	12	PI, SR & ID	ID	Nil
1	13	PI, SR & ID	ID	Nil
1	14	PI, SR & ID	ID	Nil
1	15	PI, SR & ID	ID	Nil
2	3	AN & PW CR	RC	Nil
2	4	CR, Doc, SR & ID, Art	Nil	RC
2	5	RC, Doc, SR & ID, Art	Nil	UC, Art
2	6	SR & ID, Art	Nil	SD
2	7	C4	PD	Nil
2	8	C4	PD	Nil
2	9	PD & ID	Nil	Rec
2	10	C4	PD	Nil
2	11	AN & PW CR	RC	Nil
2	12	AN & PW CR	RC	Nil
2	13	AN & PW CR	RC	Nil
2	14	AN & PW CR	RC	Nil
2	15	C2 C4	PD	Nil
3	4	CR	Nil	RC
3	5	RC	Nil	UC,Art
3	6	Art	Nil	Doc
3	7	C4	PD	Nil
3	8	C4	PD	Nil
3	9	PD & ID	Nil	Rec
3	10	C4	PD	Nil
3	15	C2C4	PD	Nil
4	5	RC	Nil	UC,Art
4	6	CR	RC	Nil
4	7	C4	PD	Nil
4	8	C4	PD	Nil
4	9	PD & ID	Nil	Rec
4	10	C4	PD	Nil
4	11	CR	RC	Nil
4	12	CR	RC	Nil
4	13	CR	RC	Nil
4	14	CR	RC	Nil
4	15	C4	PD	Nil
5	6	RC	UC,Art	Nil
5	7	C4	PD	Nil
5	8	C4	PD	Nil
5	9	PD & ID	Nil	Rec
5	10	C4	PD	Nil
5	11	RC	UC,Art	Nil
5	12	RC	UC,Art	Nil
5	13	RC	UC,Art	Nil
5	14	RC	UC,Art	Nil
5	15	C4	PD	Nil
6	7	C4	PD	Nil
6	8	C4	PD	Nil
6	9	PD & ID	Nil	Rec
6	10	C4	PD	Nil
6	11	SR & ID	SD	Nil
6	12	SR & ID	SD	Nil
6	13	SR & ID	SD	Nil
6	14	SR & ID	SD	Nil
6	15	C4	PD	Nil
7	9	C4	Nil	PD
7	11	C4	Nil	PD
7	12	C4	Nil	PD
7	13	C4	Nil	PD
7	14	C4	Nil	PD
7	15	C2C4	PD	Nil
8	9	C4	Nil	PD
8	11	C4	Nil	PD
8	12	C4	Nil	PD
8	13	C4	Nil	PD
8	14	C4	Nil	PD
8	15	C2C4	PD	Nil
9	10	C4	PD	Nil
9	11	PD & ID	Rec	Nil
9	12	PD & ID	Rec	Nil
9	13	PD & ID	Rec	Nil
9	14	PD & ID	Rec	Nil
9	15	C4	PD	Nil
10	11	C4	Nil	PD
10	12	C4	Nil	PD
10	13	C4	Nil	PD
10	14	C4	Nil	PD
11	15	C4	PD	Nil
12	15	C4	PD	Nil
13	15	C4	PD	Nil
14	15	C4	PD	Nil

Table 6: Mutant M1.

		Next state/output																	
State no.	State name	C1	C2	C3	C4	C5	C6	PI	AN&PW	RN&PW	UN&PW	PD&id	RC	CR	DOC	SR &id	art	click	C7
S1	Home	S2	S11	S12	S10/PD	S13	S14	S3/ID	-	-	-	-	-	-	-	S10	-	-	S15
S2	Login	-	S11	S12	S10/PD		S14	-	S4	S5	S6	-	-	-	-	-	-	S1	S15
S3	Registration	S2	S11	S12	S10/PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S4	Author	-	S11	S12	S10/PD	S13	S14	-	-	-	-	-	-	S8/RC	S7/UC,art	S9/SD	S8/doc	S1	S15
S5	Reviewer	-	S11	S12	S10/PD	S13	S14	-	-	-	-	-	S7/UC,art	-	S7/UC,art	S9/SD	S8/ doc	S1	S15
S6	General User	-	S11	S12	S10/PD	S13	S14	-	-	-	-	-	-	-	-	S9/SD	S8/doc	S1	S15
S7	Upload	S2	S11	S12	-	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S8	Download	S2	S11	S12	-	S13	S14	-	-	-	-	-	-	--	-	-	-	S1	S15
S9	Subscribe	S2	S11	S12	S10/PD	S13	S14	-	S4	S5	S6	S10/rec	-	-	-	-	-	S1	S15
S10	Payment mode	S2	S11	S12	-	S13	S14	--	-	-	-	-	-	-	-	-	-	S1	S15
S11	Contact us	S2	-	S12	S10/PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S12	About us	S2	S11	-	S10/PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S13	FAQ	S2	S11	S12	S10/PD	S13	S14	-	-	-	-	-	-	-	-	-	-	S1	S15
S14	Editorial Board	S2	S11	S12	S10/PD	S13	-	-	-	-	-	-	-	-	-	-	-	S1	S15
S15	Log out	S2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	S1	-

Table 7: Mutant M2.

		Next state/output																	
State no.	**State name**	**C 1**	**C 2**	**C 3**	**C4**	**C 5**	**C 6**	**PI**	**AN& PW**	**RN& PW**	**UN& PW**	**PD &id**	**RC**	**CR**	**DOC**	**SR &id**	**art**	**click**	**C 7**
S1	Home	S 2	S 11	S 12	S10/ PD	S 13	S 14	S3/ ID	-	-	-	-	-	-	-	S10	-	-	S 15
S2	Login	-	S 11	S 12	S10/ PD		S 14	-	S4	S5	S6	-	-	-	-	-	-	S1	S 15
S3	Registr ation	S 2	S 11	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	-	-	-	-	-	S1	S 15
S4	Author	-	S 11	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	-	S8/ RC	S7/U C,art	S9/ SD	S8/d oc	S1	S 15
S5	Review er	-	S 11	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	S7/U C,art	-	S7/U C,art	S9/ SD	S8/ doc	S1	S 15
S6	General User	-	S 11	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	-	-	-	S9/ SD	S8/d oc	S1	S 15
S7	Upload	S 2	S 11	S 12	-	S 13	S 14	-	-	-	-	-	-	-	-	-	-	S1	S 15
S8	Downlo ad	S 2	S 11	S 12	-	S 13	S 14	-	-	-	-	-	-	--	-	-	-	S1	S 15
S9	Subscri be	S 2	S 11	S 12	S10/ PD	S 13	S 14	-	S4	S5	S6	S10/ rec	-	-	-	-	-	S1	S 15
S1 0	Paymen t mode	S 2	S 11	S 12	-	S 13	S 14	--	-	-	-	-	-	-	-	-	-	S1	S 15
S1 1	Contact us	S 2	-	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	-	-	-	-	-	S1	S 15
S1 2	About us	S 2	S 11	-	S10/ PD	S 13	S 14	-	-	-	-	-	-	-	-	-	-	S1	S 15
S1 3	FAQ	S 2	S 11	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	-	-	-	-	-	S1	S 15
S1 4	Editoria l Board	S 2	S 11	S 12	S10/ PD	S 13	-	-	-	-	-	-	-	-	-	-	-	S1	S 15
S1 5	Log out	S 2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	S1	-

Table 8: Mutant M3.

		Next state/output																	
State no.	**State name**	**C 1**	**C 2**	**C 3**	**C4**	**C 5**	**C 6**	**PI**	**AN& PW**	**RN& PW**	**UN& PW**	**PD &id**	**RC**	**CR**	**DOC**	**SR &id**	**art**	**click**	**C 7**
S1	Home	S 2	S 11	S 12	S10/ PD	S 13	S 14	S3/ ID	-	-	-	-	-	-	-	S10	-	-	S 15
S2	Login	-	S 11	S 12	S10/ PD		S 14	-	S4	S5	S6	-	-	-	-	-	-	S1	S 15
S3	Registr ation	S 2	S 11	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	-	-	-	-	-	S1	S 15
S4	Author	-	S 11	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	-	S8/ RC	S7/U C,art	S9/ SD	S8/d oc	S1	S 15
S5	Review er	-	S 11	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	S7/U C,art	-	S7/U C,art	S9/ SD	S8/ doc	S1	S 15
S6	General User	-	S 11	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	-	-	-	S9/ SD	S8/d oc	S1	S 15
S7	Upload	S 2	S 11	S 12	-	S 13	S 14	-	-	-	-	-	-	-	-	-	-	S1	S 15
S8	Downlo ad	S 2	S 11	S 12	-	S 13	S 14	-	-	-	-	-	-	--	-	-	-	S1	S 15
S9	Subscri be	S 2	S 11	S 12	S10/ PD	S 13	S 14	-	S4	S5	S6	S10/ rec	-	-	-	-	-	S1	S 15
S1 0	Paymen t mode	S 2	S 11	S 12	-	S 13	S 14	--	-	-	-	-	-	-	-	-	-	S1	S 15
S1 1	Contact us	S 2	-	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	-	-	-	-	-	S1	S 15
S1 2	About us	S 2	S 11	-	S10/ PD	S 13	S 14	-	-	-	-	-	-	-	-	-	-	S1	S 15
S1 3	FAQ	S 2	S 11	S 12	S10/ PD	S 13	S 14	-	-	-	-	-	-	-	-	-	-	S1	S 15
S1 4	Editoria l Board	S 2	S 11	S 12	S10/ PD	S 13	-	-	-	-	-	-	-	-	-	-	-	S1	S 15
S1 5	Log out	S 2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	S1	-

A number of mutants for model FSM are generated by making use of mutation operators as described in section 4. Sample mutants for application are shown in Table P1,P2,P3.

All these FSM's are activated in their initial states by making use of test cases described in set T. with three sample mutations the methodology was able to detect some of the errors in the web application.

1. Running the application with input 'C1 AN &PW Doc' is expected to take the application to the upload state and upload confirmation and article to be produced as the output. The same is produced when test data is run on Model FSM, but when run on mutant M2, no output is generated but web application reaches upload state. That signifies the wrong output generated than expected. This is the error due to missing output function in web application.
2. Again running the application with input 'C1 RN &PW art' is expected to take the application to the download state and returning back with required document numbered as art as the output. The same is produced when test data is run on Model FSM, but when run on mutant M3, no output is generated but web application reaches download state. That signifies the wrong output generated than expected. This is again the error due to missing output function in web application.
3. Also with input 'SR & ID' application should go to the subscription page and returns back the subscription details but when mutant M1 is run with this test case it landed into payment details page and no output is generated. Again an error that represents the wrong linking of hypertext in application.

Conclusions

Mutation testing which is basically a white box testing mechanism can also be used successfully as a black box testing criteria. Mutation testing when used on specifications can be used as a magical mechanism for detecting errors of omission which could go unnoticed with any other testing criteria that emphasize on code. It can be used to ensure completeness and effectiveness of test cases in functional testing as well. Suggested Methodology could even help developers to embed complete functionalities in implementations of web applications to avoid panicky situations at users end and hence enhances the confidence in it. However it seems to be a lengthy approach to follow manually. Automation of recommended method could fasten the process of error detection.

References

Abdurazik, A., Ammann, P., Ding, W.Y., Offutt, J. 2000. Evaluation of Three Specification-based Testing Criteria. *Sixth IEEE International Conference on Engineering of Complex Computer Systems* (ICECCS2000).

Achkar, H. 2010. Model Based Testing Of Web Applications. *STANZ*, Sydney, Australia.

Andrews, A. A., Offutt, J., Alexander, R. T. 2005. Testing Web Applications by Modeling with FSMs. *Software and Systems Modeling*, 4, 326-345.

Belli, F., Budnik, C. J., Wong, W. E. 2006. Basic Operations for Generating Behavioral Mutants. *In Proccedings of 2nd Workshop on Mutation Analysis in conjunction with ISSRE*. IEEE CS, 10-18.

Beyazıt, M., Deistler, T., Gokce, N. 2010. Event-Based Mutation Testing vs. State-Based Mutation Testing – Comparison Using a Web-based System. *Lecture Notes in Informatics*, 327-332.

Black, P. E., Okun, V., Yesha, Y. 2000. Mutation Operators for Specifications. In *Proceedings of 15th IEEE International Conference on Automated Software Engineering* (ASE2000).

Demillo, R., Lipton, R., Sayward, F. 1978. Hints on test data selection: Help for the Practicing programmer. *IEEE Computer Magazine*, 11(4), 34-41.

How Tai Wah, K..S. 2003. An analysis of the coupling effect single test data. *Science of Computer programming*, 48, 119-161.

Internet world Stats usage and population statistics http://www.internetworldstats.com/stats.html accessed on 17th January, 2012

Jorgensen, P.C. 2008. Software Testing: A Craftsman's Approach, Third Edition, *Auerbach Publications*, ISBN-13: 9780849374753.

Mathur, A.P. 2008. Foundation of Software testing. *Pearson Education Publication*, ISBN:9788131707951.

Patrick, H. Z., Hall, A.V., John, H.R. 1997. Software Unit Test Coverage and Adequacy. *ACM Computing Surveys*, 29, 4.

Pinto Ferraz Fabbri, S.C., Delamaro, M.E., Maldonado, J.C., Masiero, P.C. 1994. Mutation analysis testing for Finite State Machines. *In Proceedings 5th International Symposium on Software Reliability Engineering*, 220-229.

Sabnani, K.. K.., Dahbura, A. T. 1988. A protocol test generation procedure. *Computer Networks and ISDN systems*, 15(4), 285-297.

Singh, K.., Kumar, R., Kaur, I. 2010. Testing Web applications using Finite State Machines employing Genetic Algorithm. *International Journal of Engineering Science and Technology*, 2(12), 6931-6941.

