



Analysis and Testing of High Volume Online Web-based Applications

Sanjeev Dhawan¹ and Kulvinder Singh¹

¹Faculty of Computer Science & Engineering, University Institute of Engineering & Technology, Kurukshetra
University, Kurukshetra (K.U.K) -136 119,
Haryana, INDIA.

Abstract: This paper has been designed to analyze and test the high volume Web-based hypermedia applications for evaluating their reliability and maintainability. The World Wide Web has become global communication system for delivering information and services. In addition, Web applications becoming more extensive, larger, more interactive, and more essential to the international use of computers. The technological evolution, however, is not supported by adequate Web testing methodologies. Web testing is usually carried out without following a well-defined process and lacks suitable tool support. Since the Web applications, are highly dependable, and as a ground, researcher just now beginning to understand how to model and test Web applications. The Web provides a new means to set up software and there are a number of differences between Web application and traditional methods. Web applications are installed across a network and they can reveal unusual flows of control. This research paper deals with external testing issues related to the user-level operation and software design structure for web testing. This paper modeled the web applications into Finite State Machines (FSM) and then generates the test cases based on FSM, which will provide support to the Web developers to build such systems effectively. The results obtained from the proposed method can help to identify the failure points in Web systems. The overall goal is to build effective and reliable Websites, and it can be done by discover the "best practices" of other organizations and to integrate these practices into one's own processes.

Keywords: Hypermedia, finite state machine, Web-based metric, W-method, Web log files

I. Introduction

Today's Websites are much more than simple graphically oriented hypertext or hypermedia applications and are becoming key entry points for customers and clients of an organization. However, the assessment of Websites by merely counting "hits" on a page is not an accurate measure of quality or success of Web-based hypermedia applications. Web-based hypermedia applications have great potential in areas such as education, and training to mention but a few. This research paper looks at the relative importance of Web-based metric, their design, reliability, Web site usability and maintainability index, and ranking. The results obtained from the above techniques can easily evaluate the size and other important attributes related to Web-based applications. The Web-based hypermedia applications are the non-conventional applications characterized by the authoring of information using nodes (chunks of information), links (relationship between nodes), access structures (for navigation), and its delivery over the Web. Web technologies commonly used for developing such applications are HTML, JavaScript, PHP and multimedia. The World Wide Web has created a standardized communications infrastructure that has enabled a wide range of applications, especially for business-to-business e-commerce, customer support, and entertainment.

The rapid design and deployment of web applications has been done largely in the absence of performance considerations. In addition, there have been great difficulties with forecasting site access patterns and dealing with the scalability issues. Thus, it is not surprising that Web-based applications frequently experience problems with poor reliability, long response times, and other important issues. This research paper describes finite state machines-based testing for detecting and resolving the faults in the web based applications. This present research is having three main objectives: the first is to create the Finite State Machine equivalent to the web applications via Web server log files as prime attributes, second objective is to determine the fault classes in Web application, and third aim is to generate the test cases based on FSM [3], [4], [5].

II. Concept of Finite-State Machine

A finite state machine [7] is a Six-tuple (X, Y, Q, q_0, S, O) , where: X is a finite set of input symbols also known as the input alphabet, Y is a finite set of output symbols also known as the Output alphabet, Q is a finite set of states, $q_0 \in Q$ is the initial state, $S: Q \times X \rightarrow Q$ is a next state or state transition function, and $O: Q \times X \rightarrow Y$ is an output function.

In some cases of FSM, more than one state could be specified as an initial state. In addition, sometimes it is convenient to add $F \subseteq Q$ as a set of final or accepting states while specifying an FSM. The definition of the transition function S implies that for any state q in Q , there is at most one next state. Such FSMs are also known as deterministic FSMs. In this research paper, the Nondeterministic FSMs are also concerned. The state transition function for nondeterministic FSMs is defined as which implies that such an FSM can have more than one possible transition out of a state on the same symbol. Nondeterministic FSMs are usually abbreviated as NFSM or NDFSMS or simply as NFA for nondeterministic finite automata. Another important feature of state based approach is that automata in it are used for specification, implementation and for drawing up the procedure, which is performed in terms of automata that allows to verify the propriety of FSMs functioning. This procedure could also be used for large-scale tasks and for task with difficult, smeared software logic. Let M be a formal representation of a design. Given a set of requirements R , one often constructs a design from the requirements. An FSM is one possible representation of the Web design. Let M_d be a design intended to meet the requirements in R . Sometimes M_d is referred to as a specification that guides the implementation. M_d is implemented in hardware or software. Let M_i denote an implementation that is intended to meet the requirements in R and has been derived using M_d . Note that in practice, M_i is unlikely to be an exact analog of M_d . In embedded real-time systems and communications protocols, it is often the case that M_i is a computer program that uses variables and operations not modeled in M_d . Thus, one could consider M_d as a finite-state model of M_i . The problem in testing is to determine whether M_i conforms to R .

To do so, one tests M_i using a variety of inputs and checks for the correctness of the behavior of M_i with respect to R . The design M_d can be useful in generating a set T of tests for M_i . Tests so generated are also known as black-box tests because they have been generated with reference to M_d and not M_i . Given T , one tests M_i against each test $t \in T$ and compares the behavior of M_i with the expected behavior given by exciting M_d in the initial state with test t . In an ideal situation, one would like to ensure that any error in M_i is revealed by testing it against some test $t \in T$ derived from M_d . One reason why this is not feasible is that the number of possible implementations of a design M_d is infinite. This gives rise to the possibility of a large variety of errors one could be introduced in M_i in the face of this reality, a fault model has been proposed. Given a fault model, the goal is to generate a test set T from a design M_d . Any fault in M_i of the type in the fault model is guaranteed to be revealed when tested against. The fault model is shown in the figure 1. This shows the four types of faults described by Mathur [7]: - (i) Operation error: Any error in the output generated upon a transition is an operation error. This is illustrated by the machine M and M_1 in the figure1, where FSM M_1 generates an output 0, instead of 1, when input symbol a is input in state q_0 [7]. More formally, an operation error implies that for some state q_i in M and some input symbol s , $O(q_i, s)$ is incorrect, (ii) Transfer error: Any error in the transition from one state to the next is a transition error. This is illustrated by machine M_2 in Figure 1, where FSM M_2 moves to state q_1 , instead of moving to q_0 from q_0 when symbol a is input. More formally, a transfer error implies that for some state q_i in M and some input symbol s , $O(q_i, s)$ is incorrect [7], (iii) Extra-state error: An extra state may be introduced in the implementation. In Figure 1, machine M_1 has q_2 as the extra state when compared with the state set of machine M . However, an extra state may or may not be an error. For example, in Figure 1, machine M represents the correct design. Machines M_1 and M_2 have one extra state each [7]. However, M_1 is an erroneous design, but M_2 is not because M_2 is actually equivalent to M even though it has an extra state, and (iv) Missing-state error. A missing state is another type of error. In figure 1, machine M_2 has q_2 missing when compared with machine M . Given that the machine representing the design is minimal and complete a missing state implies an error in the IUT [7].

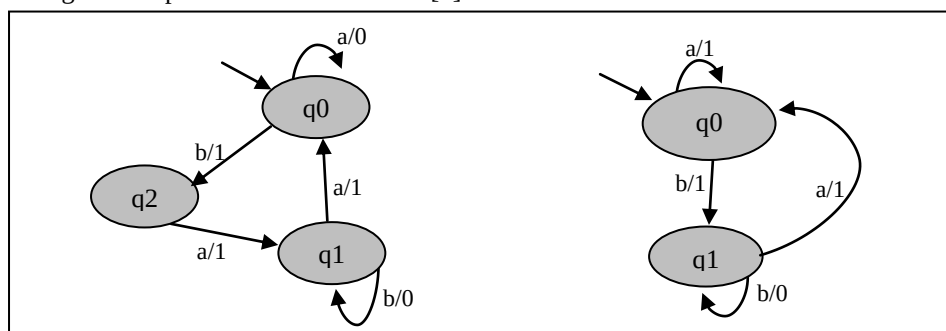


Figure 1 shows the fault model in FSM.

The above fault model is generally used to evaluate a method for generating tests from a given FSM. The faults indicated above are also known collectively as sequencing faults or sequencing errors. It may be noted that a given implementation could have more than one error of the same type. Other possibilities such as two errors one each of a different type also exist. The generation of efficient and effective test sequences is very important in Web testing. Test sequences can be generated using FSMs by applying formal methods like Transition Tours

(T-method), Unique Input Output Sequences (UIO-method), Distinguishing Sequences (D-method), Characterizing Sets (W-method) and Partial W (Wp-method). The UIO method is better than other due to: The T-method never consider state transfer faults since it does not verify the final state of a transition sequence. The W and Wp-method relies only on a reliable reset for the FSM and in reality UIOs lead to shorter test sequences than those produced using characterizing sets (W and Wp-method).

A. Testing Using The W-Method

The To test the given IUT M_i against its specification M , we do the following for each test input [7]: - (a) Find the expected response $M(x)$ to a given test input x , (b) Obtain the response $M_i(x)$ of the IUT when excited with x in the initial state, and (c) If $M(x) = M_i(x)$, then no error has been detected so far in the IUT and if $M(x) \neq M_i(x)$ implies the possibility of a fault in the IUT under test, given a correct design. Moreover, a mismatch between the expected and the actual response does not necessarily imply an error in the IUT. However, if we assume that (a) the specification is error free, (b) $M(x)$ and $M_i(x)$ have been determined without any error, and (c) the comparison between $M(x)$ and $M_i(x)$ is correct, then indeed $M(x) \neq M_i(x)$ implies an error in the design or the IUT under test.

B. Modeling Web Applications as FSMs

The modeling of the Web applications as FSMs proceeds in two parts. Part 1 builds a model of the Web application, which further includes four steps: (1) the Web application is partitioned into small modules, (2) logical Web pages (LWP) are defined, (3) FSMs are built for each module, and (4) an Application FSM is built to represent the entire Web application. Part 2 generates tests based on the model defined in Part 1 using different methods. The first step in decomposition will be to partition the Web application into modules. At the highest level, modules should be abstractions that implement functions that can be identified by users. At lower levels, modules should be cohesive software modules and Web pages that collectively work together to implement a portion of a user level function [9]. At the bottom level, modules may be individual Web pages and software modules that represent single major functions themselves. Moreover, modules can be identified from the site navigation layout, coupling relationships among the components, and design information [1], [9].

After a Web application has been partitioned into modules, the next step is to derive a Finite State Machine (FSM) for each module. First, FSMs are generated from the bottom-level clusters that only contain software modules and Web pages. Next, aggregate FSMs are constructed for high-level clusters, in which each FSM from constituent clusters is represented by a single state. Finally, an Application Finite State Machine (AFSM) will define a finite state model of the entire Web application. The Inputs to the FSMs can be of a variety of types, including text (from single digits to single lines to large files), buttons (from single button to choices from a set of buttons), links, etc. Table 1 shows a list of the types of inputs found in Web applications.

Text	Non-Text
Digit Line Email Phone URL Multilinefile	Links Buttons Radio buttons Dropdown list Check boxes

Table 1 shows the type of Inputs to the FSMs.

C. Analysis of FSM of Web Application

The FSM can be represented in form of a graph or a matrix notation. In the graph view, nodes represent the pages, and arcs represent the transition from one node to another. In the matrix representation each cell (i,j) corresponds to probability of transition from page i to page j . The construction of FSM starts with the navigational model and access logs as explained by Sengupta [8]. A sample format of IIS log file is shown in figure 2. An analysis of FSM is done using the log file of an online airline reservation system (OARS). The referrer-id and the client-id fields are considered as the basis to carry a depth first search on the access logs. This approach will separate out valid and invalid sessions. To understand, consider an application with only two independent sessions: S1 with pages (a → b → c → d → f → g) and S2 with pages (a → b → c → e → f → h). Let the access log have entries as shown in figure 3. The two valid sessions are derived when the depth first search is based on client-id fields. However, with the referrer-id field, the invalid path consisting of pages (a → b → h) is determined. The count of all such invalid sessions is determined, and the construction of FSM is done only for the valid sessions.

```
#Fields: date time c-ip s-port cs-uri-stem cs-uri-query
sc-status time-taken cs (User-Agent) cs (Referrer)
```

Figure 2 format of IIS server log file.

```
<Date and Time> <Client-id> <URL> <Referrer-id>
2012-02-07 00:00:00 201.124.225.77 a.asp
2012-02-07 00:00:02 201.124.225.77 b.asp
2012-02-07 00:00:03 201.124.225.77 c.asp
2012-02-07 00:00:05 201.124.225.77 e.asp f.asp
2012-02-07 00:00:07 201.124.225.77 f.asp a.asp
2012-02-07 00:00:06 201.124.225.77 d.asp f.asp
2012-02-07 00:00:05 201.124.225.77 f.asp g.asp
2012-02-07 00:00:10 201.124.225.77 d.asp g.asp
2012-02-07 00:00:06 201.124.225.77 f.asp h.asp
2012-02-07 00:00:06 201.124.225.77 a.asp h.asp
```

Figure 3 access log entries of IIS server for OARS.

Let us consider the example of an “Online Airline Reservation System (OARS)”, where the two sessions defined in the navigational model are: -

- (a) Session 1 (S1): It is used to “Book a seat” with pages SeatSelection.asp→SeatDetails.asp→SeatBooking.asp→SeatConfirmed.asp→SeatUnconfirmed.asp→ReservationControlNumber.asp→Payment.asp.
- (b) Session 2 (S2): It is applied to “Cancel a seat” with pages SeatSelection.asp→SeatDetails.asp→SeatBooking.asp→SeatConfirmed.asp→SeatUnconfirmed.asp→ReservationControlNumber.asp→Refund.asp.

These sessions are marked for Web pages as given in figure 4, 5 and 6. Where: a-SeatSelection.asp; b-SeatDetails.asp; c-SeatBooking.asp; d-SeatConfirmed.asp; e- SeatUnconfirmed.asp; f-ReservationControlNumber.asp; g-Payment.asp; h- Refund.asp.

Now, the following figure 4 shows the graphical view of FSM with the exit node ‘i’. The matrixes of transition probabilities for figures 4 and 5 have been shown in table 2 and 3 respectively. The matrix of table 2 considers only those sessions that have completed successfully, and the matrix of table 3 considers both successful sessions and sessions related to error nodes. For example, sum of probabilities of the paths out of the node *b* is 2 (table 2, 3 and 4) indicating that 80% of clients had either dropped out or encountered errors.

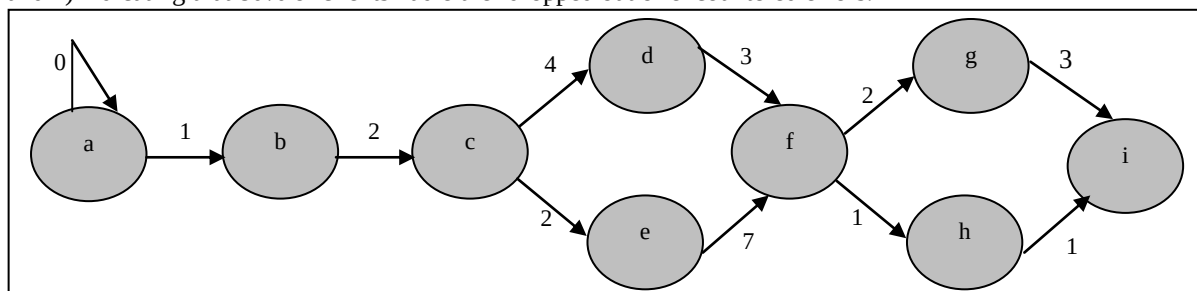


Figure 4 FSM view for OARS.

The probability of reaching a node *j* in the graph can be calculated using Markov property as described by Sengupta [8], and Wang and Tang [10]. The generalized notation of using Markov property is: -

$$N_j = N1 * P(1,j) + N2 * P(2,j) + ... + Nk * P(k,j) \quad (1)$$

Where, *k* is the number of nodes that lead to node *j*. In the OARS example of table 2 figure 4), to compute the probability of reaching the node *f* is $3 * Nd + 7 * Ne$ and probability of reaching the node *i* is $3 * Ng + 1 * Nh$, where *Na* is equal to one. In the OARS example of table 3 (figure 5), to compute the probability of reaching the error node ‘Error’ is $3 * Nd + 1 * Ng$. So, when an error node is included, then sum of all the incoming and outgoing weights of edges is increased.

	a	b	c	d	e	f	g	h	i	Sum
a	0	1	-	-	-	-	-	-	-	1
b	-	-	2	-	-	-	-	-	-	2
c	-	-	-	4	2	-	-	-	-	6
d	-	-	-	-	-	3	-	-	-	3
e	-	-	-	-	-	7	-	-	-	7
f	-	-	-	-	-	-	2	1	-	3
g	-	-	-	-	-	-	-	-	3	3
h	-	-	-	-	-	-	-	-	1	1
Sum	0	1	2	4	2	10	2	1	4	26

Table 2 matrix of transition probabilities for OARS.

In the OARS example of table 4 (figure 6), to compute the probability of reaching the error node 'Error' is $3 * Na + 3 * Nd + 1 * Ng$. Finally, from the table 2, 3 and 4; it has been observed that whenever there is an existence of new node (either error or virtual node) the transition probability of reaching that node is also increased. Therefore, to resolve such type of issues, the Web replacement policies or algorithms are developed to enhance the usability and reliability indexes of Web pages stored at different Web servers.

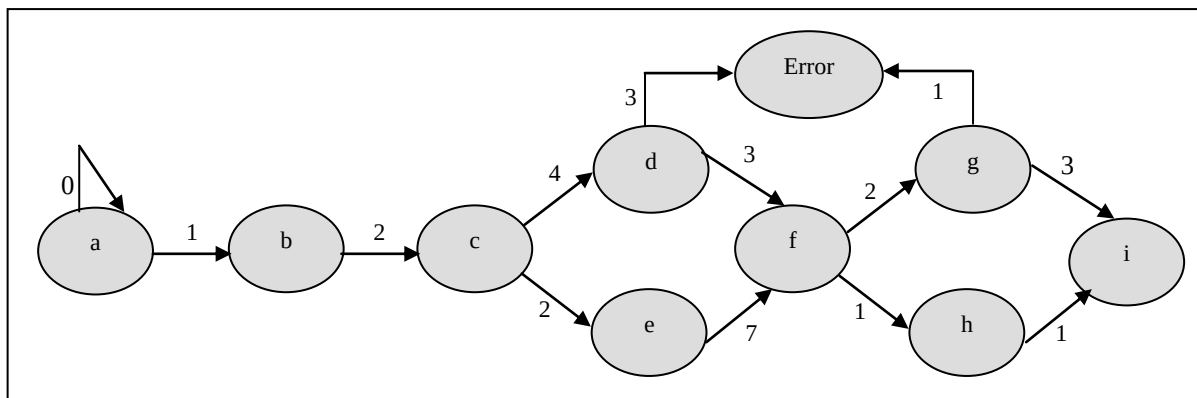


Figure 5 addition of an error node to FSM of OARS.

	a	b	c	d	e	f	g	h	i	Error	Sum
a	0	1	-	-	-	-	-	-	-	-	1
b	-	-	2	-	-	-	-	-	-	-	2
c	-	-	-	4	2	-	-	-	-	-	6
d	-	-	-	-	-	3	-	-	-	3	6
e	-	-	-	-	-	7	-	-	-	-	7
f	-	-	-	-	-	-	2	1	-	-	3
g	-	-	-	-	-	-	-	-	3	1	4
h	-	-	-	-	-	-	-	-	1	-	1
Sum	0	1	2	4	2	10	2	1	4	4	30

Table 3 matrix of transition probabilities with error node for OARS.

Now, FSM of OARS is expanded to include the failure data. To capture the failure data, the access logs are scanned for HTTP return error codes of 4xx and 5xx. Besides this, the errors from other servers are also considered. The error node *Er* is added and all the page errors are associated with this node. The matrix of transition probabilities will have an additional column to represent the error node. By considering the OARS example, the view of FSM with the addition of error node is shown in figure 5. The matrix of transition probabilities for the figure 5 is shown in table 3. The analysis of failure of FSM in terms of service-level agreements (SLA), Session Response Time (SRT), and page service time (PST) values [2]. The matrix considers only those sessions that have some error. Figure 6 shows the addition of virtual nodes to the existing figure 5. The matrix of transition probabilities for the figure 6 is shown in table 4.

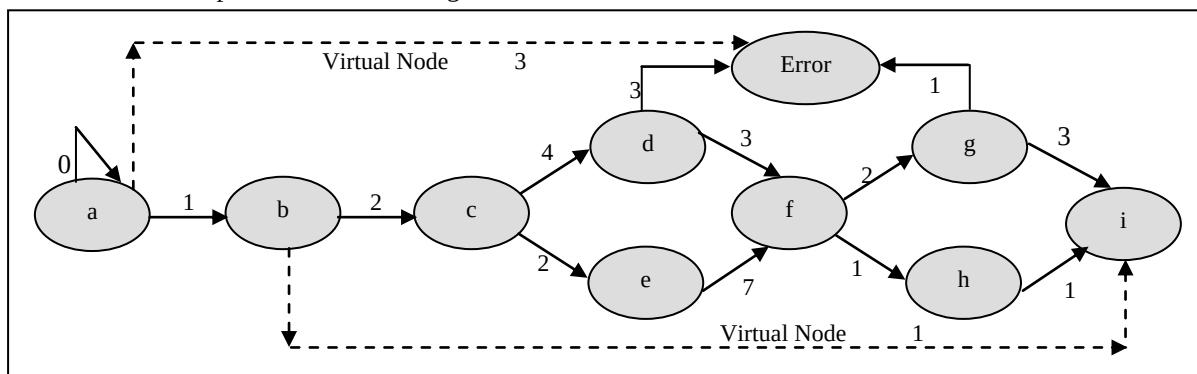


Figure 6 additions of an error and virtual nodes to FSM of OARS.

	a	b	c	d	e	f	g	h	i	Error	Sum
a	0	1	-	-	-	-	-	-	2	3	6
b	-	-	2	-	-	-	-	-	-	-	2
c	-	-	-	4	2	-	-	-	-	-	6
d	-	-	-	-	-	3	-	-	-	3	6
e	-	-	-	-	-	7	-	-	-	-	7
f	-	-	-	-	-	-	2	1	-	-	3
g	-	-	-	-	-	-	-	-	3	1	4
h	-	-	-	-	-	-	-	-	1	-	1
Sum	0	1	2	4	2	10	2	1	6	7	35

Table 4 matrix of transition probabilities with error and virtual nodes for OARS.

III. Conclusions

This paper is planned to predict and analyze the fault in Web-based hypermedia applications in terms of Finite state machine, Web page selection metrics, reliability, usability index, and error estimating function. These proposed methods gather data by the offline and online analysis of Web logs files and these methods can effectively be used for any online information systems. Firstly, an example of Online Airline Reservation System (OARS) is taken to model the web application into the Finite State Machines. Secondly, the analyze fault classes in the Web applications in the form of page faults. Finally implement different matrixes of transition probabilities. Therefore, based on the above-mentioned approach, the Web-based hypermedia applications can definitely be error-free and useful to identify the failure points in Web-based systems and improve its reliability.

IV. References

- [1] Andrews, A. A., Offutt J., and Alexander R.T., "Testing Web applications by modelling FSM", 2010.
- [2] Dhawan, S., and Kumar, R., "Effort Assessment and Predictions for High Volume Web-based Applications: a Pragmatic Approach", International ERCIM Workshop on Software Evolution, Université des Sciences et Tech. de Lille, France, 2006.
- [3] Dhawan, S., and Kumar, R., "Web Metrics for Evaluating Effort and Design of Hyperdocuments", Association for Computing Machinery New Zealand Bulletin (ACM), Vol. 3 (1), 2007, pp. 14-26 (ISSN 1176-9998).
- [4] Dhawan S. and Kumar R. "Measuring Quality Attributes of Web-based Applications Part-I: Assessment and Design" volume 23, Academic Open Internet Journal <http://www.acadjournal.com/2008/v23/part6/p7/>, 2008.
- [5] Dhawan, S., and Kumar, R., "Analyzing Performance of Web-based Metrics for Evaluating Reliability and Maintainability of Hypermedia Applications", The 3rd International Conference on Broadband Communications, Information Technology & Biomedical Applications (BroadCom 2008), Published in IEEE On-line Proceedings, 2008.
- [6] Dhawan, S., and Kumar, R., "Early Design Performance Prediction for Authoring Web-based Hypermedia Applications", 5th International Conference on Adaptive Hypermedia & Adaptive Web-Based Systems, Hannover, Germany, 2008.
- [7] Mathur, A. P., "Foundation of Software Testing", Pearson Education Publication, 2008.
- [8] Sengupta, S., "Characterizing Web Workloads- A Transaction-Oriented View", IEEE/ IFIP 5th International Workshop on Distributed Computing (IWDC- 2003), 2003.
- [9] Suny and Albany 2009, "Testing Web Applications", <http://www.coursehero.com/file/5208072/1/>.
- [10] Wang, Wen-Li, and Tang, Mei-Huei, "User-Oriented Reliability Modeling for a Web System", 14th International Symposium on Software Reliability Engineering (ISSRE), November 17-21, 2003.