Communications in Applied Analysis **10** (2006), no. 1, 85–95

USING A PETRI NET MODEL APPROACH TO WEB APPLICATIONS TESTING

Ka-Chong Ho[1] and Vincent Lau[2]

[1]Computer Studies Program
Macao Polytechnic Institute
Macao SAR, P.R. China
kcho@ipm.edu.mo

[2]Faculty of Computing,
Information Systems and Mathematics
Kingston University, UK
V.lau@kingston.ac.uk

Communicated by D.D. Bainov

ABSTRACT: Web applications are complex and growing rapidly along with the emergence of Internet and newly developed web technologies. The quality of web applications has become crucial to their success and thus should be tested thoroughly. In this paper, we describe how the all-edges testing of a web application is facilitated by creating a state transition diagram to model various interactions among web pages of each subsystem and then transform the diagram into a corresponding web Petri net machine (WPNM). Test cases can then be generated based on the reachability trees resulted from the corresponding WPNMs. We also generalize an existing statistical testing technique to all-k-paths and all-edges testing of web applications to enhance the reliability of structural testing.

AMS (MOS) Subject Classification. 62C99, 91A35, 91B06, 39A05, 39A11

1. INTRODUCTION

The ubiquity of web browsers and simple interfaces has provided a convenient way for developing distributed applications. At the beginning, web pages were simply HTML documents linked together and they were used to provide information to web-surfers across the world. web applications came to life when new technologies such as ASP (Active Server Page) and Java were introduced. web applications are now being used in many areas such as business transactions over the Internet, on-line stock trading, home shopping and banking, etc.

The complexity of web applications is rapidly increasing, as more and more services are made available over the Internet. At the same time, the quality of web

Received August 15, 2005

applications has become more critical to the integrity of the web sites and thus should be tested rigorously.

In essence, software testing is to assure the quality and proper functions of software systems. There is an established collection of literatures on both formal and informal techniques for this kind of testing in general. However, testing web applications is different from traditional software testing Pressman [12]. Software/web programmers have to cope with the dynamic, volatile, and multidisciplinary nature of the web when developing web applications.

In this paper, an approach for all-edges testing of web applications is presented. We partition a web application into several subsystems based on their functions. Having done that, we use a state transition diagram to model various interactions among web pages of each subsystem and then transform each diagram into a corresponding web Petri net machine (WPNM) with restriction for each transition to have exactly one input and one output arc. All-edges coverage criterion can then be applied upon a reachability tree resulted from the WPNM to generate test cases to facilitate the testing. The state space explosion problem could be avoided by partitioning a web application into subsystems Andrews et al [2]. We also show how an existing statistical testing technique can be generalized to a subsystem of the web application to measure the quality of test with respect to all-k-paths and all-edges coverage criteria.

This paper is organized as follows. Related works are presented in Section 2. Section 3 describes definitions and basic properties of Petri nets. Section 4 illustrates our proposed testing approach. An example is provided in Section 5 to illustrate how our approach works. Section 6 details the statistical testing process. The conclusion of this paper and future work are given in Section 7.

2. RELATED WORKS

A formal object-oriented test model has been put forwards Liu [9] for testing web applications. Traditional test models such as control flow graph, data flow graph and state chart are extended to capture the test artifacts. In particular, a Page Navigation Diagram (PND) was proposed, which is used to depict the navigation behaviour of a web application. A navigation test tree is built from a PND to detect the errors related to navigation behaviour.

A model is available that uses enhanced Finite State Machine (FSM) approach to decompose a web application into a series of finite state machines without relying on different types of diagrams Andrews and Offutt [1]. This approach to test web applications consists of two phases. In phase one, a model is built by following the four steps below:

1. A web application is decomposed into several subsystems and components.

2. Logical web pages within the application must be defined.

3. A partition FSM is built for each subsystem or component defined in step 1.

4. An aggregation FSM is built for the web application.

In phase two, tests can be generated from the model defined in phase one.

This approach assumes that the source is not available, which is similar to our approach. Unlike the object-oriented WATM Liu [9], the FSM is represented via logical, rather than physical web pages and the potential state space explosion problems are solved through partitioning and a different approach towards input description on the arcs of the FSM. The functionality testing can be done by using this approach.

A UML model of web application is proposed for web site testing Ricca and Tonella [13]. Static verification and dynamic validation are used in this approach, in which the focus is put on validating paths in a web site. Tools, namely **Reweb** and **Testweb,** are also developed to support the analysis and testing of web sites. web pages can be downloaded and analyzed by **Reweb** to build a UML model. The **Testweb** is used to generate and execute tests based on the model created from **Reweb**. The proposed UML model can be exploited to define white box testing criteria and semi-automatically generate the associated test case. The main focus of this approach is to achieve coverage adequacy.

3. DEFINITIONS OF PETRI NETS

Petri nets are classical models for modelling systems that exhibit concurrency, synchronisation and randomness. A Petri net Wang et al [19], Murata [10] is a directed bipartite graph with two node types called *places* and *transitions*. The nodes are connected via directed *arcs*. Connections between two nodes of the same type are not allowed. In graphical representation, places are represented by circles and transitions by bars or boxes.

A place P is called an input place of a transition t if there exists a directed arc from P to t, P is called an output place of t if there exists a directed arc from t to P.

A transition is called enabled if each of its input places contains 'enough' tokens. An enabled transition can fire. Firing a transition t means consuming tokens from the input places and producing tokens for the output places.

Arcs are labelled with their weights (positive integers), where a k-weighted arc can be interpreted as the set of k parallel arcs. Figure 1 shows an example of a marked Petri net.

A *marking* for a Petri net is a vector $M = (\mathrm{m}(1), \ldots . , \mathrm{m}(\mathrm{n}))$ of natural numbers, where n represents total number of places. The m(i) represents number of tokens in place P_i.

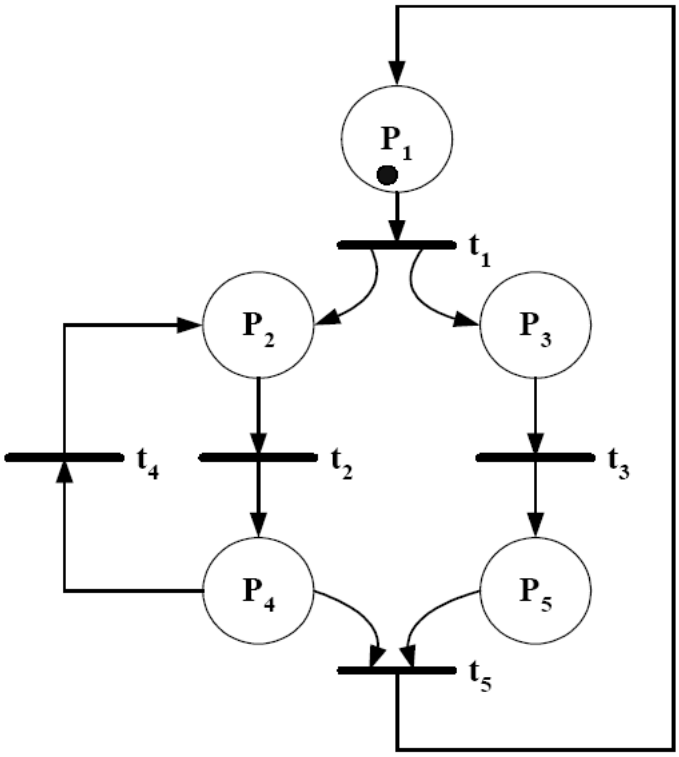

Figure 1: Example of a marked Petri net

A marking M_n is said to be *reachable* from an initial marking M_0 if there exists a sequence of firings that transforms M_0 *to* M_n. A sequence of markings $\{M_0, M_1, M_2,...\}$ and a sequence of transitions $\{t_1, t_2,...\}$ can then be defined. A reachability set $R(M_0)$ of a Petri net is a set of all markings that are reachable from M_0. The reachability set of a Petri net can be represented by a tree. The tree stops to expand if a marking that is previously defined is reached. Such reachability tree can be used to find *dead* markings, i.e., marking in which no transition is enabled.

4. THE PROPOSED TESTING APPROACH

When testing a traditional computer program, it is important to measure its internal structure that must be conformed to its design specification. Usually it can be achieved by applying some white box testing methods to the control structure of the design to derive test cases. Similar approaches as those proposed in Liu [9], Ricca and Tonella [13], Di Lucca et al [6], Elbaum et al [7] aim to generate test cases for covering link navigation paths in a web application in order to facilitate path testing. A link navigation path Di Lucca and Di Penta [5] is defined as a path from a starting page to a final page just following a given set of hyperlinks provided by the web application itself.

Most researchers have suggested that object-oriented paradigm can be used to model web applications Liu [9], Chen and Kao [3]. web pages (including client pages and server pages) and components such as Java applet, ActiveX control and Java Bean etc. can be modelled as objects and hence some object-oriented models can be built to facilitate the functional and structural testing by applying some traditional testing techniques such as data flow testing technique Liu [9]. Therefore, we extend the object-oriented class testing method proposed in Wang et al [19] together with the all-edges testing criterion to web applications. This object-oriented class testing method is used to specify method sequence specification of a class while our approach is to specify the structural aspect of a web applications in terms of paths. The following steps are involved in our testing approach:

1. A web application is partitioned into some manageable subsystems based on its functions.

2. A state transition diagram is created to model the behaviour of each given subsystem.

3. Each state transition diagram is transformed to a web Petri net machine in which each transition has exactly one input and one output arc.

4. A reachability tree is generated from each web Petri net machine.

5. All-edges coverage criterion is applied on the trees.

5. EXAMPLE

As an example we consider a simplified travel agency web application to illustrate how our testing approach works. The web application provides a platform for customers to book travels holidays on the web. The following are provided for customers to book: *Holidays*, *Flights*, *Hotels* and *Cruises*. Hence, we partition the web application into four subsystems according to the services provided. For simplicity, the *flights* subsystem is selected to go under test. The state transition diagram of the subsystem is given in Figure 2.

The graph of figure 2 can be transformed to a corresponding Petri net by simply making each state a place, and making each arc between two places a transition. The result of the transformation is shown in Figure 3.

A reachability tree can then be constructed from Figure 3. Using different coverage criteria, different sequences can be constructed for generating test cases Wang et al [19]. All-edges coverage criterion is applied upon the tree and eight sequences (paths of the tree) can be determined as shown in Figure 4. Relevant test cases can also be generated.

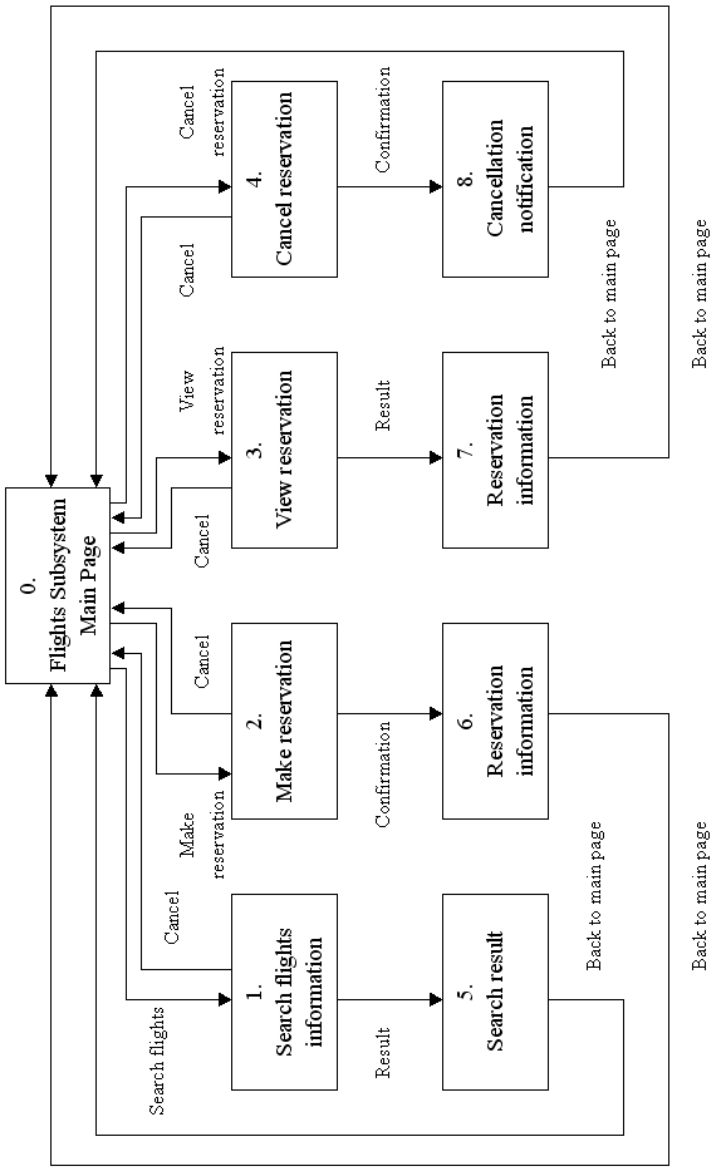

Figure 2: A state transition diagram of the *flights* subsystem

$M_0 \bullet M_1 \bullet M_0$ $(t_1 \bullet t_{13})$

$M_0 \bullet M_1 \bullet M_5 \bullet M_0$ $(t_1 \bullet t_5 \bullet t_9)$

$M_0 \bullet M_2 \bullet M_0$ $(t_2 \bullet t_{14})$

$M_0 \bullet M_2 \bullet M_6 \bullet M_0$ $(t_2 \bullet t_6 \bullet t_{10})$

$M_0 \bullet M_3 \bullet M_0$ $(t_3 \bullet t_{15})$

$M_0 \bullet M_3 \bullet M_7 \bullet M_0$ $(t_3 \bullet t_7 \bullet t_{11})$

$M_0 \bullet M_4 \bullet M_0$ $(t_4 \bullet t_{16})$

$M_0 \bullet M_4 \bullet M_8 \bullet M_0$ $(t_4 \bullet t_8 \bullet t_{12})$

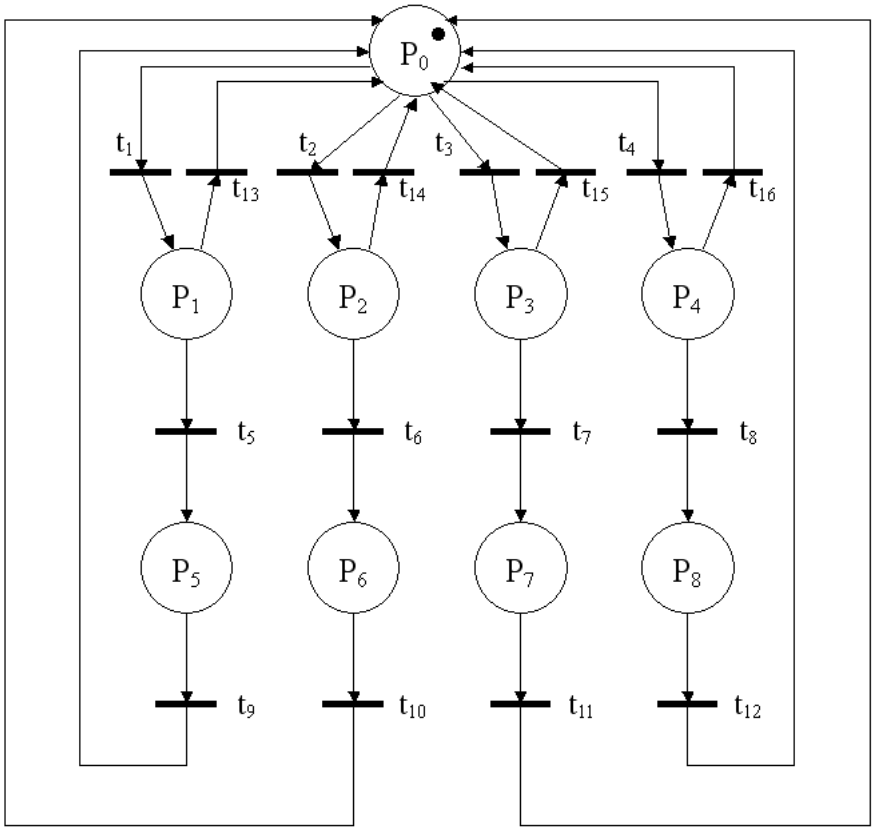

Figure 3: A WPNM model corresponding to Figure 2

6. STATISTICAL TESTING OF WEB APPLICATIONS

As outlined above, we have introduced how to use a Petri net model generated from a web application to derive test cases to satisfy the all-edges criterion. In this section, we generalize an existing statistical testing method to all-k-paths and all-edges testing of web applications to enhance the reliability of structural testing.

Statistical technique for testing web applications is proposed in Kallepalli and Tian [8], in which web usage and failure information are extracted from web logs. Usage information is used to construct Unified Markov Models Tian and Lin [17] to guide the overall testing of web pages. The failure information is used to measure the reliability of web applications.

Two tools, Reweb and Testweb, have been developed to extract a model from a web application for statistical testing Tonella and Ricca [18]. Test cases can then be automatically generated and executed to estimate the reliability of the web application under test.

An approach is proposed in Sant et al [15] to perform statistical testing by adopting statistical machine learning techniques to automatically build models from logged user data of a web application. Again, test cases can be generated for testing purpose. These statistical testing approaches can only be performed after the web application

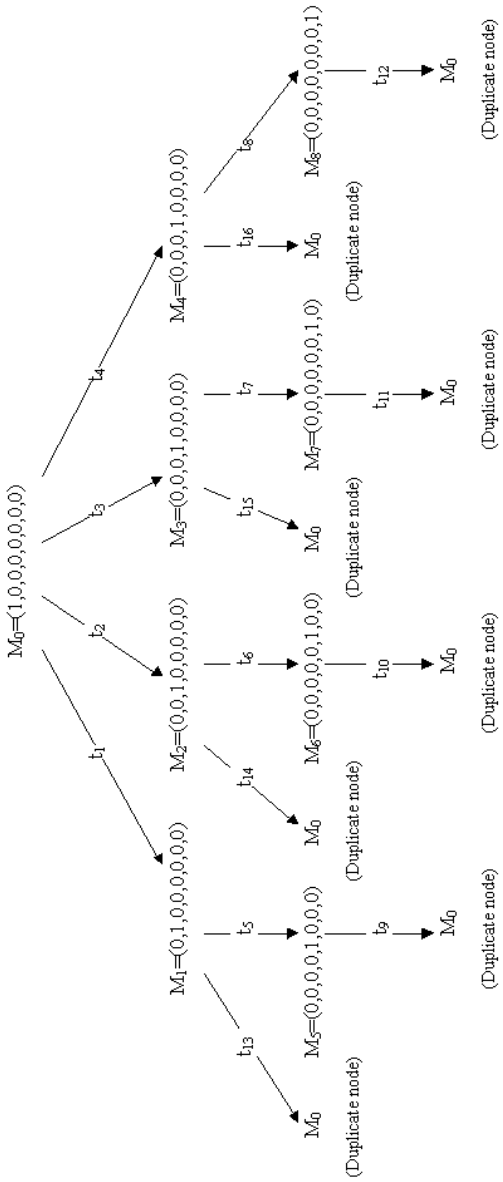

Figure 4: A reachability tree corresponding to Figure 3 for generating test cases

has been launched on the Internet. They are generally considered to be complements to structural testing of web applications. However, it is best practice to always test a web application thoroughly before putting it online.

Test cases play an important role for high quality of software testing. Inadequate test cases lead to unreliable software systems being produced. It is also necessary to measure the quality of test with respect to some coverage criteria. The notion of test quality with respect to a testing criterion for statistical testing is defined in Thévenod-Fosse [16]. It is slightly reformulated in Denise and Gouraud [4] to perform statistical testing according to any given graphical description of the behaviour of the system under test. The definition of such method is described as follows.

Let D be some description of a system under test. D may be a specification or a

program, depending on the kind of test we are interested in (functional or structural). Based on D, coverage criteria such as all-k-paths and all-edges, etc. can be defined. A coverage criterion C is covered with a probability $q_{C,N}(D)$ if each element of a corresponding set of elements $E_C(D)$ has a probability of at least $q_{C,N}(D)$ of being executed during N executions with random inputs. The quality $q_{C,N}(D)$ is a measure of the test coverage with respect to C. The following denotes the relation between test quality and the test size N:

$$q_{C,N}(D) = 1 - (1 - q_{C,1}(D))^N . \tag{1}$$

It is clear that the probability of reaching an element is one minus the probability of not reaching it N times when drawing N tests.

This method can be applied to the structural testing of web applications. The following illustrates how it could be done. When performing all-k-paths testing on a reachability tree generated from the Petri net model of the flights subsystem example shown in Figure 2, $P_{\leq n}$ and $AP_{\leq n}$denote the set of such paths and all-k-paths coverage criterion respectively. The test quality can then be obtained by the following:

$$q_{AP\leq n,N}(D) = 1 - (1 - \frac{1}{|P_{\leq n}|}))^N . \tag{2}$$

In our example, by choosing $n = 7$, there are 28 paths of length less than or equal to 7 (see Table 1).

Equation (2) becomes

$$q_{AP\leq 7,N}(D) = 1 - (1 - \frac{1}{28}))^N , \tag{3}$$

and can be used to obtain the test quality. The result is shown in Table 2. If the all-edges coverage criterion is chosen instead of all-k-paths, the result is exactly the same as illustrated in Denise and Gouraud [4].

7. CONCLUSION AND FUTURE WORK

In this paper, we extend the object-oriented class testing method proposed in Wang et al [19] to web applications. A web application could be partitioned into several subsystems based on their functions. State transition diagrams and corresponding Petri net machines with restriction can then be created based on different subsystems. All-edges coverage criterion is subsequently applied upon the reachability tree resulted from the Petri net machine of each subsystem to facilitate the

Length	Paths
2	t_1t_{13},
	t_2t_{14},
	t_3t_{15},
	t_4t_{16}
3	$t_1t_5t_9$,
	$t_2t_6t_{10}$,
	$t_3t_7t_{11}$,
	$t_4t_8t_{12}$
5	$t_1t_{13}t_1t_5t_9$
	$t_2t_{14}t_2t_6t_{10}$
	$t_3t_{15}t_3t_7t_{11}$
	$t_4t_{16}t_4t_8t_{12}$
7	$t_1t_{13}t_1t_5t_9t_1t_{13}$
	$t_1t_{13}t_1t_5t_9t_2t_{14}$
	$t_1t_{13}t_1t_5t_9t_3t_{15}$
	$t_1t_{13}t_1t_5t_9t_4t_{16}$
	$t_2t_{14}t_2t_6t_{10}t_1t_{13}$
	$t_2t_{14}t_2t_6t_{10}t_2t_{14}$
	$t_2t_{14}t_2t_6t_{10}t_3t_{15}$
	$t_2t_{14}t_2t_6t_{10}t_4t_{16}$
	$t_3t_{15}t_3t_7t_{11}t_1t_{13}$
	$t_3t_{15}t_3t_7t_{11}t_2t_{14}$
	$t_3t_{15}t_3t_7t_{11}t_3t_{15}$
	$t_3t_{15}t_3t_7t_{11}t_4t_{16}$
	$t_4t_{16}t_4t_8t_{12}t_1t_{13}$
	$t_4t_{16}t_4t_8t_{12}t_2t_{14}$
	$t_4t_{16}t_4t_8t_{12}t_3t_{15}$
	$t_4t_{16}t_4t_8t_{12}t_4t_{16}$

Table 1: 28 paths of length ≤ 7 corresponding to the reachability tree shown in Figure 4

generation of test cases. All the edges (links) of a subsystem of a web application can be tested. An existing statistical testing technique to traditional software systems is exploited to measure the quality of test with respect to all-k-paths and all-edges coverage criteria.

We have described the initial step of an ongoing research effort in enhancing the reliability of structural testing of web applications. It may also be feasible to include a set of dynamic coverage criteria of web applications defined in Sampath et al [14] to our statistical testing approach to improve the quality of our testing.

REFERENCES

[1] R. Alexander Andrews and J. Offutt, *Testing web Applications*, George Mason University, 2002.

q	0.9	0.99	0.999	0.9999
N	63	127	190	253

Table 2: Number of tests N required for a test quality q

[2] Anneliese Andrews, Jeff Offutt, and Roger Alexander, Testing web applications by modeling with FSMs, *Software Systems and Modeling*, (2004), To Appear.

[3] M. Chen and M. Kao, Testing object-oriented programs – an integrated approach, In: *Proceedings of the Tenth International Symposium on Software Reliability Engineering*, 1999, 1-4.

[4] M.-C. Gaudel Denise and S.D. Gouraud, A generic method for statistical testing, In: *Proceedings of the 15th IEEE International Symposium on Software Reliability Engineering (ISSRE 2004)*, Saint-Malo, France, November 2004, 25-34.

[5] G.A. Di Lucca and M. Di Penta, Considering browser interaction in web application testing, In: *Proceedings of the 5-th International Workshop on web Site Evolution*, Amsterdam, The Netherlands, 2003.

[6] G.A. Di Lucca, A.R. Fasolino, F. Faralli, U. De Carlini, Testing web applications, In: *Proceedings of the IEEE International Conference on Software maintenance*, Oct. 2002, Montréal, QC, Canada, 310-319.

[7] S. Elbaum, S. Karre, and G. Roothermel, Improving web application testing with user session data, In: *The 25-th International Conference on Software Engineering*, Portland, Oregon, May 2003.

[8] C. Kallepalli and Jeff Tian, Measuring and modeling usage and reliability for statistical web testing, *IEEE Transaction on Software Engineering*, **27** (November 2001), no. 11.

[9] C.H. Liu, *A Formal Object-oriented Test Model for Testing web Applications*, PhD Thesis, The University of Texas at Arlington, 2002.

[10] T. Murata, Petri nets: Properties, analysis and applications, *Proceedings of the IEEE*, **77** (1989), no. 4, 541-580.

[11] J.L. Perterson, *Petri Net Theory and The Modeling of Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[12] R.S. Pressman, What a tangled web we weave, *IEEE Software*, **17** (2000), no. 1, 18-21.

[13] F. Ricca and P. Tonella, Analysis and testing of web applications, In: *Proc. of the 23-rd International Conference on Software Engineering*, Toronto, Ontario, Canada, 2001, 25-34.

[14] Sreedevi Sampath, Emily Gibson, Sara Sprenkle and Lori Pollock, Coverage criteria for testing web applications, *Technical Report*, 2005-17, Computer and Information Sciences, University of Delaware, April 2005.

[15] J. Sant, A. Souter, and L. Greenwald, An exploration of statistical models for automated test case generation, *Workshop on Dynamic Analysis*, St. Louis, Missouri, USA, May 2005.

[16] P. Thévenod-Fosse, Software validation by means of statistical testing: retrospect and future direction, In: *Preprints 1st IEEE Working Conference on Dependable Computing for Critical Applications (DCCA-1)*, Santa Barbara, USA, August 1989, 15-22; Published in: *Dependable Computing and Fault-Tolerant Systems*, **4** (Ed-s. A. Avizienis, J-C. Laprie), Springer-Verlag, 1991, 23-50.

[17] J. Tian and E. Lin, Unified Markov models for software testing, performance evaluation, and reliability analysis, In: *Proc. of the 4-th ISSAT International Conference on Reliability and Quality in Design*, August 1998.

[18] P. Tonella and F. Ricca, Statistical testing of web applications, *Journal of Software Maintenance*, **16** (2004), no. 1-2, 103-127.

[19] C.C. Wang, W.C. Pai, D.J. Chiang, Using a Petri net model approach to object-oriented class testing, In: *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC'99),* 12-15 October 1999, Tokyo, Japan, Volume 1, 1999, 824-828.