Usage-based Statistical Testing of Web Applications

Jianhua Hao
Provenco Retail Automation
PO Box 68281 Newton
Auckland, New Zealand
0064 9 3611708
Jianhua.Hao@Provenco.com

Emilia Mendes
The University of Auckland
Private Bag 92019
Auckland, New Zealand
0064 9 3737599
emilia@cs.auckland.ac.nz

ABSTRACT
The large growth in the number of large Web applications currently developed brings concerns related to their quality, and more specifically their testing and reliability. Web application testing is still in its infancy and relies mostly upon traditional software coverage testing processes, which are highly impractical. Recent studies have looked at usage-based statistical models for testing Web applications. One of these studies, conducted by Kallepalli and Tian [8], used Unified Markov Models (UMMs), built from Web server access logs, as basis for test case selection. In addition, server error logs were also used to measure a Web application's reliability, and consequently to investigate the effectiveness of UMMs as a suitable testing mechanism. This paper describes two experiments that replicated Kallepalli and Tian's work. Our results showed that, in contrast to [8], multiple set UMMs were needed for trustworthy test case generation. In addition, our reliability assessment corroborated results from [8], confirming that UMMs seem to be a suitable testing mechanism to use to test Web applications.

Categories and Subject Descriptors
D.2.5 [**Testing and Debugging**]: Testing tools; D.2.4 [**Software/Program Verification**]: Reliability, statistical methods.

General Terms: Experimentation, Reliability.

Keywords: Statistical testing, Unified Markov Models, Experimentation, Web application testing, Web application reliability.

1. INTRODUCTION
Over the span of little over a decade the World Wide Web has grown from a small collection of Web sites to modern complex Web applications, which run large-scale software applications on distributed hardware platforms, and provide services for e-commerce, information distribution, entertainment, collaborative working, surveys, and numerous other activities.

Nowadays, many companies use a variety of Web applications and depend on these to perform their business transactions and/or to enhance their services. Companies rely upon Web applications to be dependable, secure, and able to handle large amounts of traffic concurrently. The presence of problems in any of these areas can bring an entire business to standstill and cost millions of dollars. To avoid such problems is one of the challenges of Web application testing.

ICWE'06, July 11-14, 2006, Palo Alto, California, USA.
ACM 1-59593-352-2/06/0007.

Nguyen [18] states that the primary task in Web application test planning is to target high-risk areas. It is impossible to apply traditional coverage-based exhaustive testing to Web applications because of their massive user population and interconnectedness. Therefore statistical testing based on a priority scheme is needed. Statistical testing is an application of statistical science to software intensive systems [3]. Both the goal and motivation of statistical testing are fundamentally different from normal software testing. With statistical testing, test cases are randomly generated from a usage model, which is a characterization of all possible application usage scenarios in a specific environment or situation, or for a specific user group. Thus, every possible scenario of use is traceable from the model, and could potentially be generated as a test case. Statistical testing focuses on a probability distribution obtained from the application's input domain [6]0, and makes it possible to focus testing resources on the more important aspects of the application, which reduces the overall amount of testing needed to achieve a quality product.

The outcome of the tests, performed on the randomly generated test cases, is used to estimate the application's reliability given a usage profile [27]. Thus the amount of testing resources required decreases significantly. The main benefits of statistical testing are that it prioritizes testing efforts based on a priority scheme, and allows the use of statistical inference techniques to compute probabilistic aspects of the testing process, such as reliability [2, 7, 12, 19], mean time to failure (MTTF) [4, 21], and mean time between failures (MTBF) [14].

This paper replicates Kallepalli and Tian's work [8] via two experiments where statistical testing is applied to the University of Auckland, Computer Science department (UoA/CS) Web application. Both experiments build UMMs based on access log data from two different academic terms, for different observation periods. As in [8], we also measured the reliability of the UoA/CS Web application and the effectiveness of the UMMs approach. The ultimate aim of this research is to assess the effectiveness of statistical testing and reliability analysis to ensure the delivery of reliable Web applications.

The remainder of this paper is organised as follows: Section 2 presents a literature review on Web testing, followed by a description of statistical Web testing concepts in Section 3. Section 4 presents and discusses the UMMs built for the UOA/CS Web application. Section 5 evaluates the reliability of the UoA/CS Web application and the effectiveness of UMMs. Finally, Section 6 presents our conclusions and comments on future work.

2. WEB APPLICATION TESTING LITERATURE REVIEW

2.1 Previous Studies

Wu and Offutt [30] proposed an analysis model that captures the dynamic features of Web applications. Application behavior is modeled on the principles of HTTP and HTML construction. This has advantages in terms of its independence from implementation details. The model first identifies the basic interaction unit between applications and clients as atomic sections, which contain structured information with dynamic content. Then a complete interaction is modeled using compositions and transitions representing these atomic sections. Thus a Web application W is modeled as a triple $\{S, C, T\}$, where S is the start page, C is a set of composition rules for each server component, and T is a set of transition rules. To support testing and test case generation, each Web application execution is modeled as a derivation, i.e. a sequence of interactions between clients and servers. Testing is conducted by the test cases generated from each derivation.

Wen proposed a URL-driven testing [28], which is based on the idea that scripts can be written to automatically open Web pages and their content by invoking their URLs, then success/failure messages can be returned and analyzed. There are five basic steps involved: (i) creating a basic test, (ii) converting the basic test into a URL-driven test, (iii) preparing a data table, (iv) running the test, and (v) analyzing test results.

Elbaum, Karre, and Rothermel [5] present a user session-based approach for testing Web applications. This approach differs from existing approaches in that it leverages captured user behavior to generate test cases, leading to reduced tester's intervention. Further, the results of a controlled experiment indicate that the effectiveness is comparable and likely to be complementary to more formal white-box testing approaches, such as those described in [26]. Their user session-based approach creates a model in which nodes represent Web objects (pages, frames), and edges represent relationships and interactions between objects (include, submit). They explore the notion that user session data gathered while users operate Web applications can be successfully employed in testing these Web applications, particularly as applications evolve and experience different usage profiles. The motivation behind their proposal is to minimize the cost of finding effective inputs for white-box testing.

Subraya and Subrahmanya [22] recommend a testing process that uses the concept of decomposing a Web application's behavior into testable components, which are then mapped onto testable objects. These testable objects are subject to performance testing under varied performance parameters and stress levels. As a good measure of performance testing, a set of indicators need to be defined in terms of objects, such as processor, physical disk, memory, and network traffic. Once a Web application is defined, the process of performance testing becomes restricted to testing each object or a set of objects separately. Prior to testing, the behavior of each object is captured using a "play and record" concept, and then gathered in the form of scripts known as scenarios. The scenarios are then tested with a set of performance parameters and stress levels.

Yang et al [31] present an object-oriented architecture, containing several supporting tools, that enhances traditional software testing architecture to fit common Web Application testing frameworks. The proposed architecture reuses several software patterns and architectures from traditional testing environments. In addition, a prototype of a Web application testing environment was constructed for demonstration.

Liu et al. [11] suggest a Web Application Test Model (WATM), which consists of an object model and a structure model to capture data flow information of HTML documents passed between clients and servers. Traditional data flow testing techniques are extended to Web applications. In the object model, the components of a Web application are modelled as objects and the data flow is captured by flow graphs in the structural model. Data flow test cases can be derived by a five-level approach using corresponding flow graphs based on intra-object, inter-object, and inter-client perspectives.

Kallepalli and Tian [8] recommend a statistical Web testing approach based on selective testing via a priority scheme. Their statistical Web application testing strategy consists of three steps:

1. Construct the statistical testing models based on actual usage scenarios.
2. Use these models for test case construction.
3. Analyze the test results for reliability assessment and predictions.

Prior to constructing a statistical testing model, server logs were used to gather usage information. This was the work our experiments replicated, thus further details will be provided later.

Tian et al [24] proposed a strategy to integrate existing testing techniques into a hierarchical framework, which key components were:

- Development of a high-level operational profile (OP), which enumerates major functions to be supported by Web-based applications and their usage frequencies by target customers.
- For each high-level function group, a Markov model can be constructed to thoroughly test related operations and components.
- Critical parts identified by the Markov models can be thoroughly tested using lower level models based on traditional testing techniques.

Tonella and Ricca [26] proposed a dynamic analysis technique for the extraction of a Web application model through its execution. The UML model of a Web application is extracted by analyzing the HTML code (dynamically generated). The following three heuristic criteria are used to simplify the UML model:

- Dynamic and static pages that are identical according to a character-by-character comparison are considered to be the same page.
- Dynamic pages that have identical structures, but differing texts according to their syntax trees, are considered to be the same.

Dynamic pages that have similar structures according to a similarity metric, e.g. tree edit distance on their syntax trees, are considered to be the same.

2.2 Discussion

Table 1 compares the nine studies previously presented.

Six of these nine studies [11, 28, 30, 5, 22, 31] were conducted using the traditional coverage approach, which focuses on achieving coverage adequacy and completeness. In practice it is laborious to apply such exhaustive coverage testing to Web applications. Of these six studies, [28] has only been applied and tested for a simple

test case; a prototype has been applied to [31], and only [5] and [22] have been experimentally implemented and tested, but with limitations; [5] requires an infrastructure to be set up to reproduce a Web application that resembles those found in the real world. In addition, faults also need to be seeded in the application so that testing techniques can be evaluated. In [22], different indicators for different types of Web applications need to be defined for effective testing. None of these six studies have specific tools to assist their use, which is a great problem given the growing size and complexity of Web applications, in addition to massive user volumes.

Three of the nine studies [8, 24, 26] employed a statistical testing approach, focusing on selective testing based on a probability distribution modeled on the application's input domain [6]. They all used Markov chains as usage model. The use of Markov chains to model Web usages has been justified in [10]. All three studies have specific tools to assist in their use. However, only [8] has been experimentally implemented and tested. An automated version of [24] has yet to be developed.

Finally, [26] is restrictive by its constraint to use a UML model from which to obtain states and transitions to the usage model.

The literature review suggests that UMMs usage-based statistical testing [8] seems superior to other methods for Web application testing.

Table 1 – Comparison of Web application testing methods

Study	Approach	Model	Experimentally tested?	Testing tools used	Tech-nology independent?	Merits and practical significance	Limitations
Study 1 [30]	Traditional coverage approach.	Analysis model	No	None	Yes	Due to its analytical/ rigorous nature, it is laborious to implement practically. Strong point: technology independent.	Highly analytical and rigorous; difficult to implement practically.
Study 2 [28]	Traditional coverage approach.	URL-driven navigation testing model	Yes (for a simple test case)	No specific tool used.	No	It is automated, cost-effective, maintainable, and user-friendly.	Only been tested with a simple case.
Study 3 [5]	Traditional coverage approach.	User Session Based Approach	Yes	None	Yes	Low cost compared with conventional methods.	Need to set up an infrastructure to reproduce Web applications that resemble those found in the real world; Faults need to be seeded in the application so that the testing techniques can be evaluated.
Study 4 [22]	Traditional coverage approach.	Object-driven approach	Yes	No Specific Tool used	No	This method not only ensures a structured testing methodology but also provides enhanced site quality	Different indicators need to be defined for different site types to ensure the tests are effective.
Study 5 [31]	Traditional coverage approach.	Traditional software testing architectures and software patterns	A prototype has been applied previously	ASP, ISP, JavaScript VBScript HTML Analyzer, Test case generator, recorder, composer and viewer	Yes	Handles new programming styles, induced by Web Applications.	Some aspects of Web Applications have not been covered, such as the interactions between the Web server and database
Study 6 [11]	Traditional coverage approach.	Object model, Structure model.	No	None	Yes	Unlike traditional data flow testing, this has been extended to cover HTML documents and to cross HTTP client/server boundaries.	This approach has not been validated. The prototype tool for performing this is still under development.
Study 7 [8]	Statistical testing approach.	Markov model	Yes	FastStats log file analyzer; *Analog* log analyzer	Yes	Highly practical since testing is *not* exhaustive, but selective on the basis of a priority scheme.	At present the test does not address: capturing missing Web usage information for model construction and usage time measurement.
Study 8 [24]	Statistical testing approach.	Markov model, Flat operational profile.	Yes	FastStats log file analyzer; Analog log analyzer, etc.	Yes	Strong point: hierarchical framework (merits of Study 7 also apply)	An automated version of the method is yet to be developed
Study 9 [26]	Statistical testing approach.	UML model and Markov model	No	ReWeb Spider	No	Dynamically analyzes the HTML code generated by multi-language at run-time.	In some special cases the UML model may be partial; statistical analysis and testing approaches have not been assessed experimentally.

3. THE NEED FOR STATISTICAL WEB TESTING

Statistical Web testing follows three steps, each detailed in the following sub-sections:

- Construct the statistical testing models or usage models based on actual usage scenarios and frequencies.
- Use these models for test case construction, selection and execution.
- Analyze the test results for reliability assessments and prediction, and for decision-making.

3.1 Usage Models for Statistical Web Application Testing

A usage model of an application characterizes is operation and structure [27], and corresponds to a directed graph, where nodes represent different states of the usage model, and arcs represent transactions between states. Thus, all possible uses of an application are embodied in this structure. When a probability distribution is applied to its arcs, the structure then presents the expected use of that application.

Operational profiles are used to represent usage models. An operational profile is a set of commonly used operations presented alongside its associated probabilities. An operation represents a functional requirement or characteristics of an application. According to Musa [13], an operation is a major system of logical tasks undertaken within a short duration, and its processing is substantially different from other operations. Resources are freed once an operation is completed, and control is returned to the application. A profile is a set of disjoint alternatives (of which only one can occur at a time) presented together with the probability of occurrence of each alternative [16]. The disjoint alternatives are operations. Thus an operational profile is a description of how users use an application, and can be represented using tabular or graphical format [13].

Operational profiles help guide the allocation of test cases in accordance with use, and ensure that the most frequent operations will receive more testing.

The construction of an operational profile involves 5 steps [13]:

1. Identify the initiators of operations
2. Choose a representation (tabular or graphical)
3. Create an operations "list"
4. Establish the occurrence rates of the individual operations
5. Establish the occurrence probabilities

Operational profiles can be represented using two formalisms, namely, flat operational profiles and the Unified Markov Model, both detailed below.

Flat operational profiles

Flat operational profiles describe how users use an application by listing end-to-end operations for an application and their associated probabilities [23]. When applied to Web applications, operations can represent frequently visited pages, or complete navigation patterns. Thus, a graphical representation of a flat operational profile for a Web application presents frequently visited pages, or complete navigation patterns (commonly used operations), and associated probabilities in a list, a histogram, or a tree structure. The nodes of the graph represent the frequently visited pages, or patterns. Branches represent the values (links) of the frequently visited pages, or patterns, and each value (link) has an associated probability of occurrence.

Unified Markov Models (UMMS)

Markov operational profiles model an application's operations as Markov chains, which consist of a set of states, and state transitions. Transition probabilities represent the best estimate of actual usage, and once these probabilities have been established, the Markov chain is complete [29].

There is a close resemblance between Web applications and the state transition mechanism in Markov chains [9]. We can describe the behavior of Web users as "hits and goes" [10]. Web users browse through Web pages and collect relevant information, then click links to continue their navigation. Visited pages or a collection of related pages can be considered as individual states of a Markov chain. The usage or navigation patterns, and likelihood of those navigation patterns to be followed, are captured in the state transition probabilities. In the application of Markov chain usage models to Web application testing, the complete operations or navigation patterns can be constructed by linking various states together using the state transitions (links). The probability of a complete path occurring is the product of its individual transition probabilities [1].

UMMs possess a hierarchical structure formed by a collection of Markov chains. It captures information about execution flow (control flow), transaction processing (workload creation, handling, and termination), and associated probabilistic usage information. This usage information is the key to statistical testing support, performance evaluation, and reliability analysis [23].

As an example, the top level Markov chain (see Figure 1.a), depicted by the left-most diagram, represents high-level operational units (states), associated connections (transitions), and usage probabilities (numerical quantities written beside the links). Various sub operations may be associated with an individual state and can be modeled by more detailed models, as shown in the middle (see Figure 1.b) and right Markov chains (see Figure 1.c), which depicts the expanded states for "programs" and "Masters", respectively.

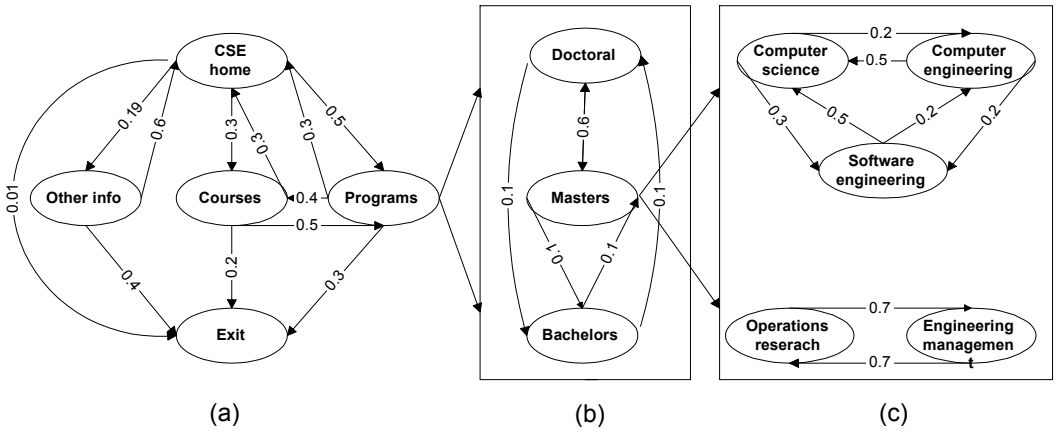

Figure 1 – UMMs (Unified Markov Models) for SMU/CSE Web pages [8]

The hierarchical structure of UMMs support selective statistical Web testing in terms of test planning and test case generation in the following ways:

- Perform exhaustive complete coverage testing for the top-level model.
- Perform selective testing for some lower-level models, which covers frequently visited sub-parts.
- Perform more selective testing for the remaining low level UMMs.

3.2 Generating Test Cases

Web application testing is based on usage frequencies derived from usage information encoded into UMMs. All possible usage scenarios are represented by UMMs, are traceable from the UMMs, and can be generated as test cases from UMMs. To generate test cases, a probability threshold needs to be established as a means of defining a "cut-off frequency" below which the corresponding navigation patterns will not be tested. The navigation patterns that exceed the probability threshold become the actual test cases to be analyzed. These patterns are formed by joining the individual states (nodes) and transitions (links) that make up the critical navigation paths. Test cases with probabilities above a certain threshold can be generated to achieve a certain testing coverage level. The threshold is used to control the number of test cases to be generated and executed. In practice, the most frequently used functions are tested first. So testing normally starts with a high threshold to cover the most frequently visited Web pages. This threshold limit can be lowered to different levels in later stages to cover more functions and achieve higher-level performance and reliability standards.

3.3 Constructing UMMs

The fundamental constituents of UMMs comprise (1) states (nodes), which are UMMs' basic elements; (2) state transitions (links), which form UMMs' structure when placed in conjunction with states; and (3) transition probabilities between states. The most fundamental task in constructing UMMs is to identify the states of use for a Web application, and their transitions.

As in [8], we built UMMs based on actual usage patterns from server logs to track Web usage and failures. The two Web server logs used were access and error logs.

Access logs record the activities of a Web server. Each time a visitor "hits" or requests a file from the Web site, the Web server records information about the "hit" to a log file. The "hit" not only registers the .html file, but also the graphics referred to in the .html page. Thus a "hit" is a request by the client/visitor (or more specifically the visitor's browser) for a resource on a Web server, and one Web page can result in many "hits".

Error logs are employed by Web servers to capture information required to analyze possible problems. They record various errors and general diagnostic data, such as when a server is started up, or is shut down.

4. APPLYING STATISTICAL WEB TESTING TO THE UOA/CS WEB APPLICATION

4.1 Introduction

Two experiments (Experiments 1 and 2) were conducted to replicate Kallepalli and Tian's work. Both experiments had to carry out the following steps [8]:

1 Construct UMMs based on actual usage scenarios derived from access logs;
2 Perform operational reliability analyses and hypothetical evaluations on the effectiveness of usage-based UMMs testing.

As in [8], both experiments used usage data from an Educational Institution's application. The Web application used in our experiments was the University of Auckland's Computer Science Web application.

To perform step 1, experiment 1 employed data from access and error logs captured by the Web server during University holidays; experiment 2 employed data from access and error logs captured during the school term. As in [8], both sets of logs covered a period of 26 days. The use of two different time periods was motivated by the belief that they would represent different usage patterns.

The results from step 1 would allow us to verify, like [8], if the UMMs created for the UoA/CS Web application represented actual usage patterns and scenarios. However, unlike [8], we also

wanted to investigate whether multiple set UMMs would be necessary to model a Web application.

Preliminary usage analysis on the access logs for the UoA/CS Web application was performed using FastStats log file analyzer, which was the same software used in [8]. FastStats wades through the collected data and stores the relevant information in log files, later translated into reports, charts, and graphs. Three of these reports are of particular importance:

1 The 'most requested pages' report, which provides the most requested pages and their associated usage frequencies.
2 The 'site entry pages' report, which details the Web pages most frequently used as entry points. This report is one of those used to build UMMs.
3 The 'tree view' report, which gives an overall graphical structure of how users move through the application. Thus, it can be used to analyze users' navigation patterns. The nodes and links of the tree view report are used as the components and connections to construct UMMs, respectively. The "hits" information is assigned as the branching probability. When UMMs represent very closely the tree structure in the 'tree view' report this indicates that UMMs adequately represent user's navigation patterns.

4.2 Constructing UMMs for the UoA/CS Web application

The use of single set UMMS or multiple set UMMS will generally be determined by the number of Web pages largely used as entry nodes in a Web application. This information is obtained using the 'site entry pages' report. [8] chose to construct single set UMMs as they had a single Web page that was used as node entry most of the time. In our case, for both experiments the 'site entry pages' report indicated that we had more than one node entry, therefore, in order to thoroughly test the UoA/CS Web application, we decided to use multiple set UMMs, one set for each different starting node. Different Web pages were used as main entry node to the top level Markov chain. UMMs were constructed based on the information provided by the 'tree view' report and 'site entry pages' report generated by FastStats.

Analysis of the multiple set UMMs for experiments 1 and 2, and the reports generated by FastStats, confirmed that the UMMs truly represented users' navigational patterns. Consequently, these results suggest that UMMs provide a reliable basis for representing usage behavior.

4.3 Generating Test Cases for the UoA/CS Web application

As previously mentioned, to generate end-to-end test cases, a "cut-off frequency" needs to be defined by establishing a probability threshold below which the corresponding navigation patterns will not be tested. We employed the same threshold used in [8].

Once UMMs have been created for a Web application 'a' and used to generate test cases, the next step is to assess how effective UMMs were in detecting failures. This assessment is carried out by measuring the reliability of 'a'.

5. MEASURING THE RELIABILITY OF THE UOA/CS WEB APPLICATION AND EFFECTIVENESS OF UMMS

Software reliability is the probability of failure-free operation of a computer program for a specified time in a specified environment [15]. A range of reliability models exists to assess software reliability. Thus, software reliability can be measured as the rate of occurrence of failures (ROCOF) [20], i.e. the current rate at which failures are occurring; the mean time to failure (MTTF) [20], i.e. the time expected to elapse between now and the next failure; and the mean-time-between-failure (MTBF) [20], i.e. the time expected to elapse between one failure and the next. As in [8], we used the Nelson model [17] to assess the application. This is a quantitative, time related model based on the number of errors found during testing and the amount of time it took to discover them. If the number of errors exceeds *f*, for a total of *n* "hits", the estimated reliability *R* according to the Nelson model is obtained as:

$$R = 1 - (f/n) \quad (1)$$

Error analysis

The error log analysis provided by FastStats was not adequate to our needs, thus utility programs written in Java were developed to extract detailed information of different types of errors from the error log. Our utility programs performed similar functionality to those used in [8].

In relation to the most common types of errors, results were as follows:

- "Permission denied" and "file does not exist" were the most dominant errors.
- In [8] the error "File does not exist" occurred 10 times more often than "permission denied" errors. In contrast, our experiments showed that "permission denied" errors occurred more often than "File does not exist" errors.

It is important to note that "permission denied" errors were not taken into account by the reliability analysis in [8], as it was believed that this type of error did not represent a failure. However, such assumption may be too simplistic since there are instances when permission is incorrectly denied and thus would represent a failure corresponding to a fault that needs to be fixed.

Given a set of UMMs, the overall reliability can be calculated based on the information of reliabilities of individual nodes and transitions, and state transition probabilities. The overall reliability of the UoA/CS Web application was assessed using Nelson's model and MTBF.

The reliability assessment for the UoA/CS Web application for Experiment 1 yielded the following reliability and MTBF:

$$\mathrm{Re}liability = 1 - 13349/886391 = 98.49\% \quad (2)$$

$$MTBF = 886391/13349 = 66.40\ "hits" \quad (3)$$

These values mean that, based on the data gathered during *Experiment 1*, the UoA/CS Web application was 98.4% reliable and there was, on average, one error for every 66.40 "hits".

The reliability assessment for the UoA/CS Web application for Experiment 2 yielded the following reliability and MTBF:

$$\text{Re}liability = 1 - 28668/1766161 = 98.38\% \quad (4)$$

$$MTBF = 1766161/28668 = 61.61 \text{"}hits\text{"} \quad (5)$$

These values mean that, based on the data gathered during *Experiment 2*, the UoA/CS Web application was 98.38% reliable and there was, on average, one error for every 61.61 "hits".

Our results confirmed those of [8], where a linear relationship was observed between usage intensity and failure counts (see Figures 2 and 3). This means that the number of errors encountered per day varies in proportion to a Web application's usage.

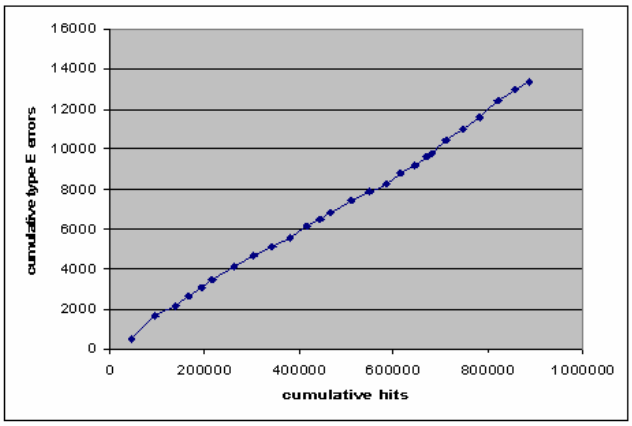

Figure 2 – Cumulative errors versus cumulative "hits" (Experiment 1)

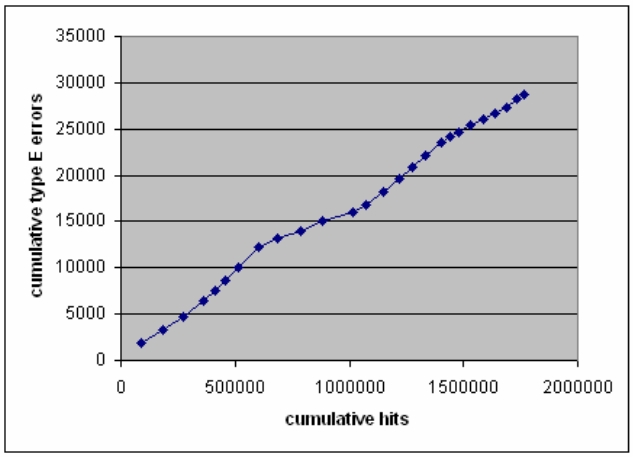

Figure 3 – Cumulative errors versus cumulative "hits" (Experiment 2)

The results from experiments 1 and 2 show that during the school term, UoA/CS Web application has a slightly lower reliability (R= 98.38%, MTBF = 61.61) than during University holidays (R = 98.49, MTBF = 66.40). However, both reliability and MTBF values are very similar, and overall suggest that the UoA/CS Web application is stable and reliable.

6. CONCLUSIONS

This paper described two studies that replicated C. Kallepalli and J. Tian's UMMs approach by applying UMMs usage-based statistical Web application testing to the UoA/CS Web application.

The results of our usage analyses suggest that users' navigation patterns and associated frequencies, and Web application behaviors are encoded into UMMs. We confirmed that UMMs are suitable and adequate to be used as usage models for statistical Web application testing. We also showed that it was essential to construct multiple set UMMs for Web application testing rather than single set UMMs, as proposed in [8]. This stresses the need to carefully decide on whether to construct single set UMMs, or multiple set UMMs, since it can affect the level of thoroughness applied to subsequent Web application testing, and the reliability analysis.

To measure the reliability of the UoA/CS Web application, we calculated the site reliability and MTBF by feeding usage data and failure data into Nelson's model, and obtained reliability of 98.49% and MTBF = 66.40 "hits" for experiment 1, and reliability of 98.38% and MTBF = 61.61 "hits" for experiment 2.

Both values indicated that the UoA/CS Web application was stable and reliable.

As part of our future work we will replicate our experiments using other educational Web applications.

7. REFERENCES

[1] Carroll, C. T. The cost of poor testing: A U.S. government study (part 2). EDPACS, The EDP Audit, Control and Security Newslstter. 31, 2, 1-16, 2003.

[2] Cheung, R. C. A user-oriented software reliability model. IEEE Trans. Software Eng., vol. SE-6, Mar. 1980.

[3] Chillarege, R. Software Testing Best Practices. IBM Research Technical Report. http://www.chillarege.com/authwork/papers1990s/TestingBestPractice.pdf (last accessed) 2004.

[4] Currit, P. A., M. Dyer, H. D. Mills. Certifying the correctness of software. IEEE Trans. Software Eng., vol. SE-12, no. 1, pp. 3-11, Jan. 1986.

[5] Elbaum, S., S. Karre, G. Rothermel. Improving web application testing with user session data. Proc. of the 25th Int'l Conf. on Software Engineering, pp. 49 – 59, May, 2003.

[6] Gouraud, S. D., A. Denise, M. C. Gaudel, B. Marre. A new way of automating statistical testing methods. IEEE Software, 2001.

[7] Hamlet, R.. Testing software for software reliability. Tech. Rep. TR 91.2, rev. 1, Dept. of Comput. Sci., Portland State Univ., Portland, OR, USA, Mar. 1992.

[8] Kallepalli, C., J. Tian. Measuring and modeling usage and reliability for statistical web testing. IEEE Trans. Software Eng. vol. 27, no. 11, pp. 1023-1036, 2001.

[9] Karlin, S., H.M. Taylor. A first course in stochastic processes, second ed. New York: Academic Press, 1975.

[10] Li, Z., J. Tian. Testing the suitability of Markov chains as Web usage models. Proc. of the 27th Annual intl. computer software and applications conf., pp. 356-362, 2003.

[11] Liu, C.H., D. C. Kung, P. Hsia. Object-based data flow testing of web applications. Proc. of the First Asia-Pacific Conf. on Quality Software, pp. 7 –16, 2000.

[12] Miller, K. W., L. J. Snell, R. E. Nooman, S. K. Park, D. M. Nicol, B. W. Murrill, J. M. Vias. Estimation the probability of failure when testing reveals no failure. IEEE Trans. Software Eng., vol. 18, pp. 33-43, Jan. 1992.

[13] Musa, J.D. Software Reliability Engineering. McGraw-Hill, 1998.

[14] Musa, J. D. A theory of software reliability and its application. IEEE Trans, Software Eng., vol. SE-1, pp. 312-321, Aug.1975.

[15] Musa, J. D., A. Iannino, K. Okumoto. Software reliability. McGraw-Hill Pubilshing Co., 1990.

[16] Musa, J. D.. Operational profiles in software reliability engineering. IEEE Software. vol. 10, no. 2, pp. 14-32, 1993.

[17] Nelson, E.. Estimating software reliability from test data. Microelectronics and Reliability, vol. 17, no. 1, pp. 67-73, 1978.

[18] Nguyen, H. Q.. Testing applications on the Web. New York: John Wiley & Sons, Inc, 2001.

[19] Pamas, D. L., A. J. Van Schouwen, S. P. Kwan. An evaluation of safety-critical software. Commun ACM, vol. 23, pp. 636-648. June 1990.

[20] Rook, P.. Software Reliability Handbook, London, New York, 1990.

[21] Shooman, M. l.. Software Engineering: Design, Reliability, and Management. New York: McGraw-Hill, 1983.

[22] Subraya, B. M., S. V. Subrahmanya. Object driven performance testing of Web applications. First Asia-Pacific Conf. on Proc. of Quality Software, pp. 17-26, no. 30-31, Oct. 2000.

[23] Tian, J. and E. Lin. Unified Markov Models for Software Testing, Performance Evaluation, and Reliability Analysis. 4th ISSAT Int'l Conf. on Reliability and Quality in Design, Seattle, Washington, August, 1998.

[24] Tian, J., L. Ma, Z. Li, A. G. Koru. A hierarchical strategy for testing web-based applications and ensuring their reliability. Proc. of the 27th Annual International Computer Software and Applications Conference, 2003.

[25] Tian, J., S, Rudraraju, Z, Li, Evaluating Web software reliability based on workload and failure data extracted from server logs, IEEE transactions on software engineering. Vol. 30, no.11, November 2004.

[26] Tonella, P., F. Ricca. Dynamic model extraction and statistical analysis of web applications. Proc. of the Fourth Int'l Workshop on Web Site Evolution, 2002.

[27] Trammell, C. Quantifying the reliability of software: Statistical testing based on a usage model, Proc. of the second IEEE int'l symposium on software engineering standards, Canada, Aug. 1995.

[28] Wen, R. B. URL-Driven automated testing. Proc. Of the second Asia-Pacific Conference on Quality software, 2001.

[29] Whittaker, J. A., M. G. Thomason. A Markov chain model for statistical software testing. IEEE Trans. Software Eng., vol. 20, no. 10, pp. 812-824, 1994.

[30] Wu, Y., J. Offutt. Modeling and testing web-based applications. Technical Report ISE-TR-02-08, 2002.

[31] Yang, J. T., J. L. Huang, F. J. Wang, W. C. Chu. An object-oriented architecture supporting Web Application testing. 23rd Int'l Computer Software and Applications Conference, pp:122–127, 27-29, Oct.1999.