Evaluating Reliability-Testing Usage Models

Bo Wan
The school of EECS
University of Ottawa
Ottawa, Canada
Bwan080@uottawa.ca

Gregor v. Bochmann
The school of EECS
University of Ottawa
Ottawa, Canada
bochmann@site.uottawa.ca

Guy-Vincent Jourdan
The school of EECS
University of Ottawa
Ottawa, Canada
gvj@ eecs.uottawa.ca

***Abstract* - Testing the reliability of an application usually requires a good usage model that accurately captures the likely sequences of inputs that the application will receive from the environment. Markov usage models and their variations have been found to be well suited for generating test cases that are statistically close to what the application is expected to receive when in production. In this article, we study the specific case of web applications. We present an evaluation method for estimating the accuracy of various reliability-testing usage models. The method is based on comparison between observed users' traces and traces inferred from the usage model. Our method gauges the accuracy of the reliability-testing usage model by calculating the sum of goodness-of-fit values of each traces and scaling the result between 0 and 1.**

Keywords: Web applications, Usage model evaluation, Reliability testing, Markov chains.

I. INTRODUCTION

In the past two decades, the problem of improving software quality has attracted a lot of research interest. One quality attribute of importance is software reliability. For a material system, reliability is usually defined by the expected time of operation after which the system will fail. In the case of a software system, it can be defined by the expected number of usages before it will fail. A usage, in this context, may be a request provided by the environment, or a complete usage session, for instance in the case of an application with an interface to a human user.

Many formal reliability estimation techniques have been discovered and developed: reliability growth models [1], operational profile testing [2] and statistical usage testing [3]. Statistical usage testing is used to validate the developed software based on the intended usage, and provides reliability measurements as well. However, in practice, the coverage provided by the test cases cannot be exhaustive. Thus the purpose of a realistic reliability testing campaign cannot be to execute every possible test case. In this context, it is important to test first those behavior patterns that occur most frequently under normal operating conditions. This idea has been applied with great success to certain large software projects: Google was for example able to deliver an internet browser, Chrome, that was remarkably reliable from its first release, not necessarily because it was tested against more web pages than the other browsers, but because it was tested against the web pages that Google knew people were most looking at[1].

Several usage models proposed in the past two decades can be applied to simulate the operational environment of the system under test for reliability estimation. In [4], a tree-based structure is used to represent the collection of paths inferred from a log file in order to present the users' web surfing pattern. First-order Markov chain models are also widely used in reliability testing [5], [6], [7]. In 2000, Borges and Levene proposed a new Markov model based on extracting user navigation patterns by using a Hypertext Probabilistic Grammar model (HPG) and N-grams [8]. Later, they presented a new method that uses clustering in a way that enables the model to represent higher-order conditional probabilities [9]. More recently, we proposed a hybrid tree-like Markov usage model. Assuming that most users' usage sessions start with popular pages, we constructed a model based on a modified tree that captures the most frequent behaviors, while adding also a Markov chain that captures infrequent behaviors [10].

In this paper, we provide a method to measure the accuracy of a usage model as compared with the users' real usage history in order to answer the following questions encountered in reliability testing: (1) Given several usage models, which one captures best the users' operational profile and should thus be used for reliability estimation, in order to improve the accuracy of software reliability testing? (2) If the usage model is parameterized, how can the values of the parameters be optimized? We focus our discussion around Web applications. Our method evaluates the accuracy of a model by taking into account the covariance differences between the observed transition visiting frequencies and the model-implied transition visiting frequencies, given all possible users' web visiting trails. We store the observe user's behavior and the model-implied behavior in two matrices capturing the probabilities of choosing a given page as the next page, given the path that was followed to reach the current page. Clearly, the better the usage model captures the user's behaviors, the more similar the two matrices are. The covariance difference criterion used is chi-square (χ^2). We present a way to determine the extent to which the observed user usage behaviors fit the Markov usage models. In order to strengthen our experimental results, we use the k-fold cross validation testing strategy [11].

[1] See http://www.google.com/googlebooks/chrome/, page 10 for a graphical illustration.

This paper is structured as follows. Section 2 presents an overview of Markov usage models and their applications in reliability testing. Section 3 presents a detailed description of our proposed method for evaluating the accuracy of Markov usage models. In Section 4, we detail our experimental results and discuss a new way to optimize parameter-based usage models. In Section 5, we give our conclusions and present our plans for future research.

II. RELIABIILTY-TESTING USAGE MODELS

A conventional definition of software reliability is the probability that software will not fail in a specified number of usages in a given operational environment. The operational environment is characterized by an input distribution and a description of the data that potentially will be processed by the software [12]. Such operational environment can be perfectly described by a usage model which includes the probability density function that defines the likelihood that any element in the domain is chosen for execution; it is the input distribution for running the software. Several published techniques for estimating the reliability of a system are developed based on these concepts (such as [12], [13], [14]).

In 2002, Sayre and Poore applied a first-order Markov Chain usage model to estimate single-use reliability which was defined as the probability of the software executing a randomly selected input without failure [15]. In their usage model, the states represent modules or functions of the system under test, and arcs among states represent state correlations of the user behavior, the states "*Invoke*" and "*Terminate*" denote the start and the end states of the user behavior. Hence, a use or a test case is an executed sequence of actions from "*Invoke*" to "*Terminate*" generated by a random walk on the Markov chain usage model. The mean and the estimated variance of the software reliability are estimated for the sample generated by simulation of the usage model. More recently, an industrial tool named MaTeLo also used Markov chains to model usage profiles, to generate test cases, and to debug and estimate the software reliability [6], [7], [16]. In 2011, Sprenkl et al. presented an empirical study of the generation of test sequences from higher-order Markov Chain usage model for web applications testing [17]. The test sequences are generated from random walks on the usage model, following the probability distribution of the model transitions. They compared how using different order Markov Chain impacts the resulting model size and accuracy and the characteristics of the generated test sequences. In this paper, we provide a generalized approach to measure the accuracy of various Markov usage models.

In Web applications, a first-order Markov model captures the page-to-page transition probabilities: $p(x2|x1)$ where $x1$ denotes the current page and $x2$ denotes one of the next pages reachable from $x1$. Such low-order Markov models cannot capture behavior where the choice of the next page to be visited depends on "history", that is, on how the current application state was reached. An example is shown in Figure 1: In the snippet of a traditional Markov usage model (b), we see that there are three ways to reach state 3 (from state 1, from state 2 and from state S), and that from state 3, there is 30% chance to go to state 4, and 70% chances to go the state 5. However, looking at the provided application state sequences, we can see that users reaching state 3 from state 1 never go to state 4 afterwards. What is shown in the first-order Markov chain is misleading. Thus, some test sequences generated by a first-order Markov chain cannot represent realistically certain usage patterns of the web application[2]. The estimated reliability is inevitably biased by those misleading test sequences. Since it is reasonable that most Web applications involve such history-dependent behavior, accurate models of user behavior cannot be obtained with first-order Markov chains [Peter]. The same problem is also discussed by Deshpande and Karypis [Deshpande].

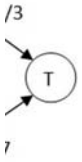

Figure 1 An example of a first-order Markov Chain model: (a) a collection of application state sequences and (b) the corresponding first-order Markov chain model

In the past decades, several published higher-order Markov usage modeling in data mining and pattern recognition are also suitable for reliability estimate. An N-gram navigation model has been explored by Borges and Levene in 2000 to extract user navigation patterns by using a Hypertext Probabilistic Grammar model structure (HPG) and N-grams [8]. In their work, an N-gram captures user behavior over a subset of N consecutive pages. They assume that only the N-1 previous pages have a direct effect on the probability of the next page selected. To capture this, they reuse the concept of "gram" taken from the domain of probability language learning [16]. Consider, for example, a web site composed of six states {A1, A2, A3, A4, A5, A6}. The observed application state sequences are given in Table 1 (Nb denotes the number of occurrences of each sequence).

[2] Similar criticisms can of course be made of first-order Markov models for non Web applications.

TABLE 1. A COLLECTION OF APPLICATION STATE SEQUENCES

Application State Sequences	Nb
A1-A2-A3	3
A1-A2-A4	1
A5-A2-A4	3
A5-A2-A6	1

A bigram model is established using first-order probabilities. That is, the probability of the next choice depends only on the current position and is given by the frequency of the bigram divided by the overall frequency of all bigrams with the same current position. In the example of Table 1, if we are interested in the probabilities of choices from application state A2, we have to consider bigrams (sequences including two application states) that start with state A2. This includes the following: Segment A2-A3 has a frequency of 3, and other bigrams with A2 in their current position include the segments A2-A4 and A2-A6 whose frequency are 4 and 1, respectively; therefore, p(A3|A2)=3/(3+4+1)=3/8. It is not difficult to see that the 2-gram model is a first-order Markov chain, the first-order Markov usage model. The second-order model is obtained by computing the relative frequencies of all trigrams, and higher orders can be computed in a similar way. Figure 2 shows the 3-gram model corresponding the sessions in Table 1.

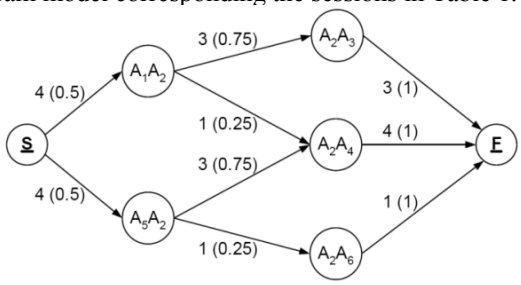

Figure 2. 3-gram model corresponding to the sessions given in Table 1

Subsequently, the same authors showed in 2004 how to use higher-order Markov models in order to infer web usage from log files [8]. They propose to duplicate states for which the first-order probabilities induced by their out-links diverge significantly from the corresponding second-order probabilities. Take Table 1 again as example. Consider state 2 and its one-order probability *p(A3|A2)*=3/8, and its two-order probability *p(A3|A1A2)*=3/4. The large difference between *p(A3|A2)* and p(A3|A1A2) indicates that coming from state A1 to state A2 is a significant factor on the decision to visit A3 immediately afterwards. To capture this significant effect, they split state A2 as illustrated in figure 3. A user-defined threshold defines how much the first and second order probabilities must differ to force a state splitting. A k-means clustering algorithm is used to decide how to distribute a state's in-links between the split states.

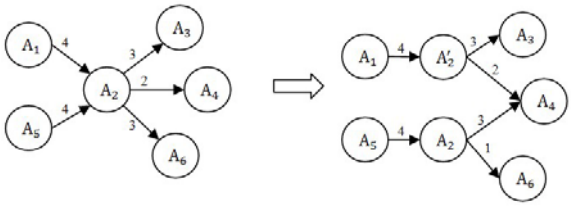

Figure 3 an example of the cloning operation in dynamic clustering modeling

More recently, we have presented a new method that can be used to create an accurate statistical usage model for reliability testing of web applications from log files [10]. Since for web applications the choice of the next page to be visited depends on the "history", the user's operational profile in the context of the previous pages visited can be presented as a conditional probability $p(s_i|s_1, s_2, \dots, s_{i-1})$where $s_1, s_2, \dots, s_{i-1}$ is the visiting trail of past web pages is. Using the definition of conditional probability, we see that $p(s_i|s_1, s_2, \dots, s_{i-1})$ equals

$$p(s_i|s_1, s_2, \dots, s_{i-1}) = \frac{p(s_1, s_2, \dots, s_{i-1}, s_i)}{p(s_1, s_2, \dots, s_{i-1})} \quad (4)$$

Using the chain rule, the probability of a visiting trail is given by:

$$p(s_1, \dots, s_{m-1}, s_m) = p(s_1) \prod_{i=2}^{m} p(s_i|s_1, \dots, s_{i-1}) \quad (5)$$

In practice we always keep user visiting sessions as well as conditional probabilities of visiting sessions in a tree structure. Let $c(s_1, s_2, \dots, s_{i-1}, s_i)$ denote the frequency count of the trail, then the estimate of the conditional probability by the frequency counts is (6):

$$p(s_i|s_1, s_2, \dots, s_{i-1}) = \frac{c(s_1, s_2, \dots, s_{i-1}, s_i)}{c(s_1, s_2, \dots, s_{i-1})} \quad (6)$$

Although the tree-structured Markov usage model contains the full user behavior information, it is quite impractical for the use in reliability estimation, mainly due to its very large size. Our hybrid tree-like Markov usage model preserves the strength of the tree-structured Markov usage model while providing high coverage and good scalability. The method uses a tree structure to preserve statistically significant information of the user behavior, as gathered from the log files. The initially very large tree (shown on a very small sample on Figure 4.b) is reduced in three steps: first, frequency pruning removes the branches that are almost never followed. The pruning is controlled by a parameter, called "frequency threshold" θ. When the calculated conditional probability of a branch is lower than the frequency threshold, the branch is cut. Then, a test known as the "Cochran criterion" is used to remove states that do not carry reliable statistical information. This Cochran criterion states that in order to apply the test of independence, at most 20% of the possible alternatives should have fewer than six instances in the sample set [18]. To avoid the loss of the model's coverage, the states removed during these two steps are merged into a first-order Markov chain model (called the "lower Markov chain") that captures infrequent behaviors. Following the Cochran criterion, the tree model after frequency pruning (θ =5%) and Cochran criterion pruning is illustrated in Figure 4.c.The pruned tree is further reduced

through merging of model states corresponding to the same application states and on which user behavior is statistically similar. For example, in Figure 5.a, the model states 1.a and 1.b denote the same application state with different "histories". If the users' behavior is very similar at some application state even though the trails by which they reached this application state are different, the model state should be merged. The judgment of similarity is made by an independence test. After merging, the resulting "tree", which is called the "upper tree", contains the most frequent behaviors, which are statistically significant. Figure 4.d shows the model after merging by independence test with significance level 0.05. In practice, the resulting hybrid Markov usage model is drastically smaller than the original tree of sequences, but still contains all the significant behavioral and coverage information.

ı table a

n table a

Figure 4 An example of hybrid tree-like Markov usage model

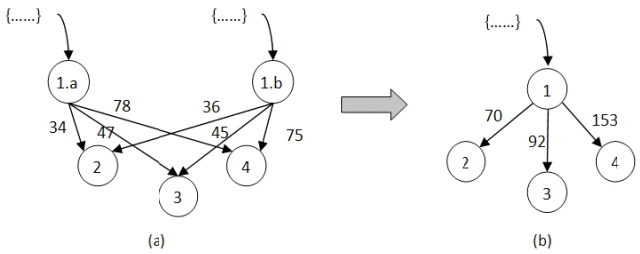

Figure 5 An example of merging two model states

III. MODEL EVALUATION METHOD BASED ON COVARIANCE ANALYSIS

As reviewed in Section 2, lower-order or higher-order Markov models can be used to capture software usage patterns. In other words, they all can be used to simulate the operational environment of a system for reliability testing. However, the following question arises: Which model can best represent the real operational behavior of the users? – Or in other words: To what extent are the test cases generated by a usage model similar to the observed user behavior as documented in the execution sequences recorded in the application log files? These questions will have to be answered by the software engineer who wants to select a usage model that is to be used for estimating the reliability of the system through reliability testing. Therefore, finding a statistically significant usage model has a practical and substantive meaning.

a. Observed data and model-implied data

A good Markov usage model contains the users' usage information and accurately captures the probabilities of these usages. In other words, if we observe the users' behaviors under the same conditions, the behavior suggested by the usage model should not be significantly different from the behavior observed with real users. For example, Table 2 is a collection of observed application state sequences from users' usage, also called test sample, and the usage model under test is a first-order Markov usage model (Figure 1.b) obtained from a training sample (Figure 1.a).

TABLE 2. AN OBSERVED SAMPLE OF APPLICATION STATE SEQUENCES

Application State Sequences	Nb
S-1-3-5-T	8
S-3-4-T	2
S-2-3-4-T	3
S-2-3-5-T	9

Our observation sample of Table 2 suggests that users reaching state 3 after the state sequence s-1-3 always choose

state 5 afterwards, which can be expressed by the conditional probability $p(5|s,1,3) = 1$. The observed frequency of choosing 5 after visiting trail s-1-3 is $c(5|s,1,3) = 8$. In the following, we call this the "observed values". On the other hand, the usage model implies that the probability of choosing state 5 with the previous visiting trail s-1-3 is 70%. Counting from observation sample, the $c(s,1,3) = 8$. Thus, the model indicated frequency of choosing 5 after the visiting trail s-1-3 is $c(5|s,1,3) = p(5|s,1,3) * c(s,1,3) = 0.7 * 8 = 5.6$. In the following, we call this the "model-implied values". Due to the significant difference between the observed and model-implied values, we can say that the usage model does not capture the user's behavior after visiting s-1-3.

Based on this simple example, if we want to evaluate the whole model's fitness, we need to analyze the differences between observed and model-implied values under all conditions. Assume $T = s_1, s_2, \dots, s_{i-1}, s_i$ is a trail observed in the test sample, the model implied frequency value, $c(s_i|s_1, s_2, \dots, s_{i-1})$, is calculated by formula (7), where $p_m(s_i|s_1, \dots, s_{i-1})$ is the conditional probability from the model and $c(s_1, \dots, s_{i-1})$ is the frequency of prefix subsequence $s_1, \dots, s_{i-1}$ in the test sample.

$$c(s_i|s_1, \dots, s_{i-1}) = p_m(s_i|s_1, \dots, s_{i-1}) * c(s_1, \dots, s_{i-1}) \quad (7)$$

All observed values and model-implied values are stored in two: the sample matrix (S) and the expected matrix (E). For the example of Table 2, the matrices S and E are shown in the table 3

TABLE 3. AN EXAMPLE OF OBSERVED VALUES AND MODEL IMPLIED VALUES

	s	s-1	s-2	…	s-2-3	…
1	8	0	0	…	0	…
2	12	0	0	…	0	…
3	2	8	12	…	0	…
4	0	0	0	…	3	…
5	0	0	0	…	9	…

(a) S-Observed values

	s	s-1	s-2	…	s-2-3	…
1	8.8	0	0	…	0	…
2	8.8	0	0	…	0	…
3	4.4	8	12	…	0	…
4	0	0	0	…	3.6	…
5	0	0	0	…	8.4	…

(b) E-Model-implied values

S[i][j] represents the observed frequency of going to state i when the prefix j was followed. Similarly, E[i][j] represents the model-implied frequency of going to state i when the prefix j is followed. Note that S contains all the information we can get from the observation sample.

b. Logic of chi-square test

The classic statistical method to evaluate whether a model fits the observation is through covariance analysis and the chi-square test. Borges and Levene also used the chi-square test to estimate the predictive power of higher-order Markov models [19]. The χ^2 value is given by (8)[3]

$$\chi^2 = \sum\sum \frac{(S_{ij} - E_{ij})^2}{E_{ij}} \quad (8)$$

A significant χ^2 value relative to the degrees of freedom indicates that the observed and model-implied matrices differ. A non-significant χ^2 value indicates that the two matrices are similar; indicating that the usage model accurately represents the usage pattern. Whether a χ^2 value is significant or non-significant is determined by the χ^2 distribution and its null hypothesis. If the χ^2 value is larger than χ_i^2 given by the corresponding χ^2 distribution and a certain significant level, we reject the hypothesis, that is, the model under test does not represent the user's behavior. Otherwise, we say the usage model fits the user behavior represented by the observation sample.

However, in practice the, chi-square test has limitations:

1. Some basic assumptions underlying the chi-square test may be false and the distribution of the statistics may not hold when these assumptions are violated [20].

2. A chi-square test offers only a dichotomous decision strategy implied by a statistical decision rule and cannot be used to quantify the degree of model fitness along a continuum [21]

3. The chi-square test of model fitness can lead to erroneous conclusions. Since the chi-square test is a direct function of the sample size, the probability of rejecting the model increases as the sample size increases, even when the model is minimally false [22].

c. Logic of the Goodness-Of-Fit Index test

George Box said "All models are wrong, but some are useful" [23]. As we discussed above, the chi-square test may reject any model when the sample size is large enough. But some of the rejected models are still quite useful. Our goal should not be to decide whether the model is "correct" or "wrong", but to describe the extent to which the model captures the data. In particular, we want to be able to know whether one model fits the data better than another one.

Markov usage models can be used to create a series of nested models capturing more and more "history". It starts with an independent usage model, which does not capture any correlations between states at all. In this case, the user behavior as described by a model that does not depend on any factors, not even the current state. Then, the first-order Markov usage model introduces the first-order conditional probabilities: it captures a behavior where the choice of the next operation depends on the current state Then, higher-order Markov usage models can capture correlations between states that are further and further apart in the "history" leading to a more and more detailed usage pattern.

[3] If the model does not expect a sequence contained in the observed sample, the corresponding entry in the observation matrix S is a non-zero value, yet the model-expected value is zero. In this case, we correct χ^2 to $(S_{ij} - E_{ij})^2/1$.

Finally, the series of usage models ends with an ideal Markov usage model which is able to describe exactly the user behavior and all correlations in the observed usage behavior. In other words, the ideal Markov usage model fits all observations. Thus, the difference between the values of the observation matrix S and the matrix E of the ideal Markov usage model is 0 in all situations. Consequently, we can position any possible Markov usage model on a scale ranging from 0 to 1 representing the so-called Goodness-of-Fit Index (GFI), where the independent model is the reference of the worst usage model (GFI = 0) and the ideal Markov usage model is the best one (GFI = 1). The equation to find the position of any "proposed" usage model on this scale is given by (9) [22]

$$GFI = \frac{\chi^2_{independent} - \chi^2_{proposed}}{\chi^2_{independent} - \chi^2_{ideal}} \quad (9)$$

where $\chi^2_{independent}$ is the chi-square value of the independent model, χ^2_{proposed} denotes the chi-square value of the proposed model and χ^2_{ideal} expresses the chi-square value of ideal model (thus χ^2_{ideal} is always 0 by definition).

We note that the GFI value has no statistical meaning, therefore it is not easy to provide a meaningful interpretation. For instance, what does it mean if the GFI value of a model is 80%? – Experience is required to associate some meaning the various possible GFI values. Bentler and Bonett claim that a GFI value of more than 90% indicates the model fits the observed data well [22]. This shortcoming of the GFI value comes with an advantage: Since there is no statistical meaning, we do not have to worry about the basic assumptions on the statistical distributions (see limitation (1) at the end of Section III (b)). In particular, we do not have to worry about the "Cochran criterion" which is often applied to the chi-square values for usage sequence of low frequency.

In practice, statistical accuracy is not the only criterion that is relevant to evaluate a usage model. The model size is also of importance. In fact, provided that we have access to a training sample that is large enough, the tree model will have a GFI of 1. However, this model cannot really be used because of its very large size. Therefore, in our model evaluation, in order to chose a model for reliability testing, we would tend to select a model that has fewer states among the ones that have a good enough GFI (say above 90%).

IV. EXPERIMENT

a Goodness-of-fit Index

We conducted experiments with a real data set from a web site called Bigenet (http://www.bigenet.org). Bigenet is a genealogy web site allowing access to numerous registers – birth, baptism, marriages, death and burials – in France. The data set is organized as a list of visiting sessions from the access log files of the web server and the functional model of the application. We had at our disposal the access log files for the period from September 2009 to September 2010. Table 4 presents a summary of the characteristics of the visiting sessions during this period.

TABLE 4. SUMMARY STATISTIC FOR THE DATA SET FROM BIGENET

Characteristics	Bigenet
Num. of Application States	30
Num. of Request	638546
Num. of Sessions	88666
Num. of Application State Sequences	27107
Ave. Session length	7.20
Max. Session length	199

In order to avoid the estimate bias, we used k-fold cross validation: we split the whole sample S into k sub-samples $s_1, \ldots, s_k$ of approximately equal size. The usage model is trained and tested k times. For each $t \in \{1, \ldots, k\}$, we use all sub-samples except s_t to train model, then use s_t to test the model. In this experiment, we split our sample set S of 88666 visiting sessions into three randomly selected sub-samples (folds). This insures that the usage model is trained and tested with large enough data sets. Thus, each usage model is trained and tested three times, and the GFI reported in Table 5 is the average.

We tested four different usage models. For each one, we report the GFI value as well as the number of states in Table 1.

TABLE 5. SUMMARY RESULT OF THE GOODNESS-OF-FIT INDEX TEST FOR FOUR DIFFERENT USAGE MODELS

	Goodness-of-fit index	Model size
First-order MCUM	72.84%	32
Hybrid tree-like MUM	88.69%	950
Optimized hybrid tree-like MUM	91.98%	898
Tree model	98.69%	225066

The four models shown in Table 5 are the following. (1) The first-order Markov chain usage model, (2) the "hybrid tree-like Markov usage model" presented in [10] with a frequency pruning threshold of 5%, (3) an "optimized hybrid tree-like Markov usage model" where a branch is pruned when there are less than 25 observations for it (instead of having less than 5% of the alternatives), and (4) the tree model containing all branches of the training sample. We note that the first-order Markov chain usage model has a relatively low GFI. Thus, it is not very suitable to simulate software's operational environment of reliability testing. Although the hybrid tree-like Markov usage model has a much better GFI, it is still below 90%, which shows that it could be improved. Tracing the issue, we found that using a percentage to prune the low-frequency branches in the tree may lead to over-pruning. Since the hybrid tree-like Markov usage model is a parameter-based model, we were able to optimize the pruning parameter (in this case by setting it to a count of 25) to achieve high accuracy while maintaining a small model size.

As shown in Table 5, we note that the accuracy of the optimized hybrid tree-like Markov usage model has an improved GFI value (91.98% instead of 88.69%) while the model size is reduced by more than 50 states. The GFI value of the tree model is nearly 100%. The difference of 1.31%

results from the difference between the training sample and testing samples. This difference decreases as the sizes of the training and testing samples increase. In this case, the tree model, which approaches the ideal model, has an GFI value that is 6.71% better than the optimized hybrid tree-like Markov chain usage model, however, the size of this model is much larger (by a factor 250).

b Comparing the Generated Test Sequences

The goal of our evaluation of reliability-testing usage model is of course to be able to select a model from which we can generate test sequences that fit user's behavior well, and that is not too large. In order to see visually how well the test sequences generate from the model fit, we selected the first-order MCUM and the Optimized hybrid tree-like MUM models and used a random walk on the model to produce sample sequences.

In this experiment, we randomly selected 29,546 sessions out of the 88,666 to train our models. We used another 29,546 sessions from the remaining set as reference, and generated 29,546 sessions from both models. We sorted the sequences using radix-sort to have a consistent and regular ordering of the sequences in all three sets. We then map the sorted sequence sets to the images (shown in Figure 6) by assigning colors to different application states. Figure 6 shows the difference between the three sets, where white place means empty.

As explained in Section IV a., the GFI of first-order MCUM is 72.84%, while the GFI Optimized hybrid tree-like MUM is 91.98%. As expected, the graph obtained from the sequences of Optimized hybrid tree-like MUM (middle) closely matches the graph obtained from the reference set (top), while the graph obtained from the sequences of first-order MCUM (bottom) is much more different. This provides a visual confirmation of our results.

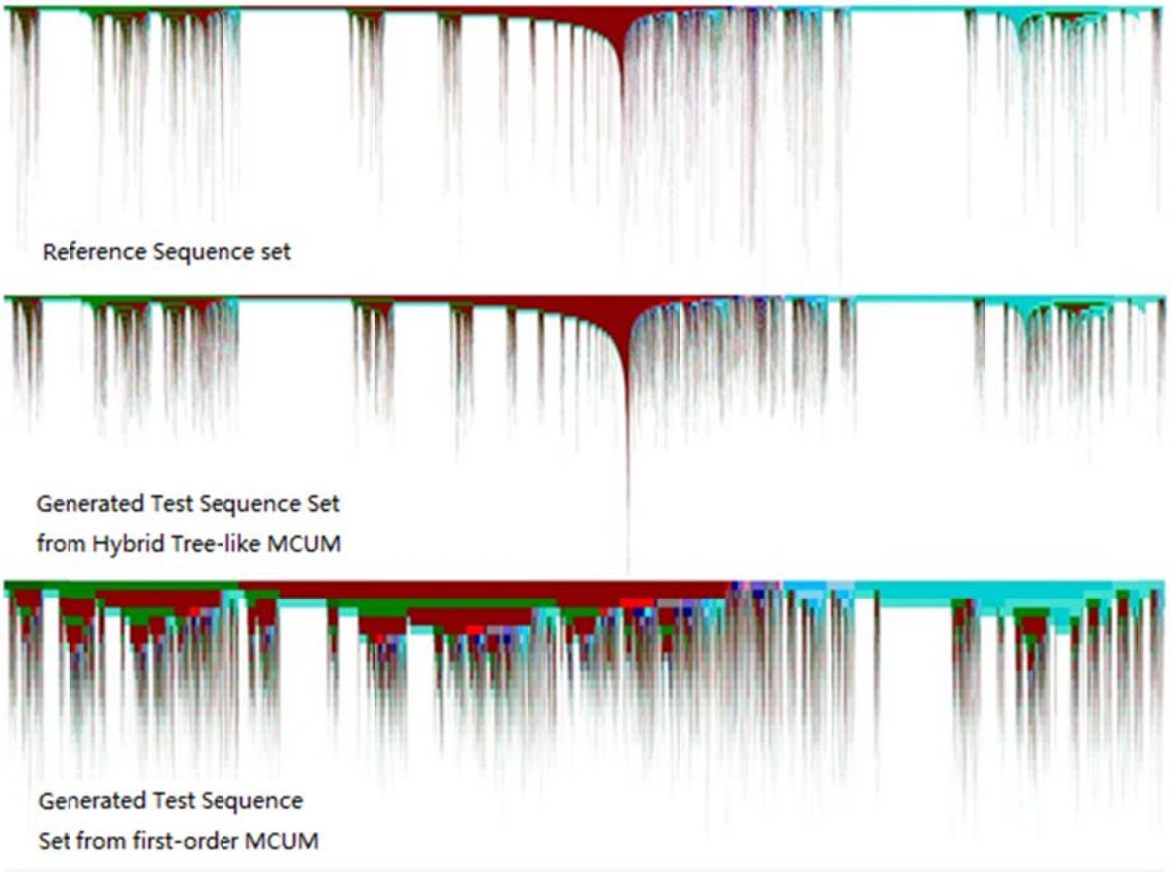

Figure 6 Generated Test Sequences Sets from different usage Models

c Selecting Best Model Parameters

In parameter-based usage models, using different parameters with the same training rules leads to different models. In this section we use our hybrid tree-like Markov usage model to illustrate how the Goodness-of-Fit Index can be used to optimize parameter-based usage models. As explained in Section II, the hybrid tree-like Markov usage model has parameters (frequency-pruning and Cochran criterion) that can lead to a usage model that has a very much reduced "upper tree" and injects more information into the "lower Markov model". Therefore, the frequency-pruning thresholds play a crucial role for model accuracy and size. In our previous paper, we introduced these parameters but did not discuss how to set them.

The elimination of low-frequency branches in the upper tree can be done via two parameters: the percentage threshold θ and the count threshold c. When the conditional probability of a branch is lower than θ and/or the frequency of a branch is lower than c, the branch is cut.

TABLE 6. THE TRENDS OF MODEL SIZE AND MODEL ACCUARCY WITH DIFFERENT PARAMETERS

(a) GFI matrix

θ\c	N/A	500	200	100	50	25	0
N/A	72.84	87.96	89.87	90.84	91.47	91.98	92.73
0.20	83.14	90.96	91.42	91.94	92.23	92.41	92.73
0.15	87.72	91.13	91.58	92.04	92.31	92.46	92.73
0.10	87.98	91.48	91.9	92.17	92.36	92.5	92.73
0.05	88.69	91.82	92.21	92.36	92.48	92.57	92.73
0.00	92.73	92.73	92.73	92.73	92.73	92.73	92.73

(b) Model size matrix

θ\c	max	500	200	100	50	25	0
1.00	32	87	160	278	485	898	2051
0.20	191	756	980	1201	1406	1694	2051
0.15	520	917	1123	1360	1537	1769	2051
0.10	589	1121	1331	1532	1677	1835	2051
0.05	944	1416	1633	1738	1824	1927	2051
0.00	2051	2051	2051	2051	2051	2051	2051

Table 6 shows the how model accuracy (a) and the model size (b) change with different parameters values. The rows show how these values evolve with c for a fixed θ and the lines show the other way around. Each branch is pruned if its conditional probability is lower than θ and the branch frequency is lower than c. The matrices show us that with θ and c decreasing, the accuracy of the model increases but so does its size. One can then choose parameters that yield a good enough accuracy (usually above 90%) with an acceptable model size.

V. CONCLUSION

In this paper, we have presented a method to assess the accuracy of various Markov usage models. The goal is to help test designer to select a practical usage model for web application reliability testing. The method uses the chi-square value to measure the distance between the probability distributions of predicted by the usage model and the probabilities observed in a testing sample. We use the Goodness-of-Fit Index (GFI) to position the proposed model on a scale ranging from a worst usage model to an ideal model that fits all observations. We can position any Markov usage models on this scale to reflect its relative accuracy. Therefore, testers can now compare different Markov usage models and selected the one that yields an acceptable accuracy and has an acceptable size. It is also possible to use this approach to optimize the values of model parameters in order to find the best usage model instance.

In contrast to some empirical studies on the same question (such as [17]), we provide a statistical view of model fitness and accuracy that can be applied to all Markov usage models. Compared with previous studies based on classical chi-square tests to estimate the model's accuracy (such as [19]), our method overcomes the following two weaknesses. First, the result of a chi-square test depends on the test sample size. When the test sample size is small, the chi-square test tends to accept the hypothesis that the model accurately captures the user's behavior. On the other hand, when the test sample is large, the chi-square test tends to reject the hypothesis. Our method reduces this impact of the sample size. Second, the chi-square test is a dichotomous decision based on the chi-square distribution relative to the degrees of freedom; it determines the confidence that the model accurately represents the behavior of the testing sample. The Goodness-of-Fit Index positions the accuracies of a model within a continuous closed interval which makes it easy to compare the accuracy of different usage models. Thus, it is easy to determine the best model or most suitable model for the simulation of the operational environment in reliability testing.

REFERENCES

[1] IEEE Reliability Society, "IEEE Recommended Practice on Software Reliability", IEEE Std 1633-2008, New York, 27 June 2008.

[2] J.D.Musa, "Operational profiles in software-reliability engineering", IEEE Software, 10(2):14–32, Mar. 1993.

[3] R.C.Cheung, "A user-oriented software reliability model", IEEE transactions on software Engineering, SE-12(1):118–125, Mar. 1980.

[4] S .Schechter , M. Krishnan and M.Smith, "Using path profiles to predict HTTP requests". Comput Netw ISDN Syst 30:457–467, 1998

[5] J.A.Whittaker and M.G.Thomason, "A Markov Chain Model for Statistical Software Testing," IEEE Trans. Software Eng., Vol. 20, No. 10, pp.812–824. 1994

[6] H.Le Guen, R.Marie and T.Thelin, "Reliability Estimation for Statistical Usage Testing using Markov Chains". In ISSRE '04: Proceedings of the 15th International Symposium on Software Reliability Engineering, pages 54-65, Washington, DC, USA. IEEE Computer Society, 2004.

[7] W.Dulz and F.Zhen, "MaTeLo—statistical usage testing by annotated sequence diagrams, Markov chains, and TTCN-3", In Proceedings of Third International Conference On Quality Software (QSIC'03), IEEE, 2003.

[8] J.Borges and M.Levene, "Data Mining of User Navigation Patterns" WEBKDD'00, pp. 92-112, 2000

[9] J.Borges and M.Levene: "A dynamic clustering-based Markov model for web usage mining". In CoRR: the computing research repository. cs.IR/0406032, 2004.

[10] G.Bochmann, G-V.Jourdan and B,Wan. "Improved Usage Model for Web Application Reliability Testing", The 23rd IFIP Int. conference on Testing Software and Systems (ICTSS'11), 2011.

[11] R.Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, International Joint Conference on Artifical Intelligence", (IJCAI), 1994.

[12] K.W.Miller, et. al., "Estimating the Probability of Failure When Testing Reveals No Failures", IEEE Transactions on Software Engineering, Vol 18, pp 33-42. 1992.

[13] T.A.Thayer, M.Lipow, and E.C.Nelson, "Software Reliability" (TRW Series of Software Techn., Vol. 2). New Yourk: North-Holland, 1978

[14] J. Neyman, "Outline of a theory of statistical estimation based on the classical theory of probability," Phil. Trans. Roy. Soc., London A., vol. 236, p. 333, 1937.

[15] K.Sayre and J.Poore, "A Reliability Estimator for Model Based Software Testing", 13th Int'l Symp. Software Reliability Engineering, pp. 53-63, 2002.

[16] A.Feliachi and H.Le Guen, "Generating transition probabilities for automatic model-based test generation", Third International

Conference on Software Testing, Verification and Validation, pp. 99-102, 2010

[17] S.Spernkle, L.Pollock and L.Simko, "A study of Usage-Based Navigation Models and Generated Abstract Test Cases for Web Application", Fourth International Conference on Software Testing, Verification and Validation, 2011

[18] R.E.Walpole and R.H.Myers, "Probability and Statistics for Engineers and Scientists", Fifth Edition, published byMacmillan publishing company (1993)

[19] J.Borges and M.Levene: "Testing the predictive power of variable history web usage". Soft Comput 11:717-727, 2007.

[20] P.M.Bentler, "Comparative fit indexes in structural models." Psychol Bull. 1990 Mar;107(2):238-46.

[21] L.Hu, & P.M.Bentler, "Evaluating mode fit". In R. H. Hoyle (Ed.), Structural equation modeling: Concepts,issues and applications (pp. 76–99). Thousand Oaks, CA: Sage 1995.

[22] P.M.Bentler and D.G.Bonett, "Significance Tests and Goodness of Fit in the Analysis of Covariance Structures" Psychological Bulletin 1980 Vol. 88, No. 3 588-606.

[23] George E. P. Box "Empirical Model-Building and Response Surfaces", co-authored with Norman R. Draper, p. 424, ISBN 0471810339, 1987.