Tunghai Journal 38 : 65-78
July, 1997

A Quadratic Programming Approach to Usage Modeling for Software Reliability Certification

Wen-Kui Chang*

Abstract

In software usage testing, a software usage model has to be developed first to characterize a population of all possible uses of the software in order to generate statistically a sample of test cases, and then to estimate software reliability accordingly.

However, it is usually difficult to completely specify all one-step transition probabilities in a Markov chain to represent the software usage model due to uncertainty of future expected use patterns. In this paper, a summary of software usage testing will first be presented, and then formulation of usage modeling with mathematical programming is discussed and demonstrated. The problem will be solved by a quadratic programming method. The advantage of this approach is its ease with the sensitivity analysis, and the timing saving in building up the model once some state parameters are further ascertained.

Keywords: software usage testing, quadratic programming, Markov chain, software reliability certification

Introduction

Recent research on related literature justifies model-based statistical usage testing. For instance, Musa [Musa93]reports a benefit-to-cost ratio of 10 or greater in developing and using operational usage models. Furthermore, Walton [Walton95]claims that the failures occurred most frequently in field use will be found early in the test cycle under statistical testing based on a software usage model.

The main benefit of software usage testing is that it makes use of statistical inference techniques to compute probabilistic aspects of the testing process, such as reliability, or mean time to failure.

Specifically, the rationale of software usage testing lies in the fact that the failures occurred most frequently in practical use will be found early in the test

*TungHai University, Dept. of Computer & Information Sciences

cycle. Besides, random testing based on usage distributions ensures that on average the most frequently used operations receive the most testing.

In this paper, software usage testing is carried out through a Markov model for software usage. Ideally, all the parameters of the usage model are well established using information obtained from various sources, including both the software's intended function and usage patterns of previous versions or prototypes of the software. However, it often happens that complete information about the usage probabilities may not be available from current environment. This research explores the improvement on usage modeling for the sake of insufficient information.

In the following, key concepts on software reliability certification, and usage testing will first be presented and discussed. An issue on usage modeling is then pointed out in the section C. A solution by the quadratic programming approach is proposed in the following section with a demonstration example.

Software Reliability Certification

Functionally, software is deemed reliable on its use and intended performance. To certify a software product, a product measure should describe how well it will operate. Software reliability is then estimated by observing its execution behavior and failure characteristics as well under the software's intended operating environment..

However, software use must first be modeled as a random process in which a use may then be selected according to some use distribution. The mean time to failure (MTTF) is computed as the average number of uses between failures. Time is measured as the number of uses or test cases. Furthermore, software reliability may be considered as the probability that software will perform correctly according to its specification for a randomly selected use [Poore93].

As in hardware analysis, software reliability is mathematically related to its MTTF by the following formula:

$$MTTF = 1/(1 - reliability) \qquad (1)$$

Note that in this sense software reliability is slightly different from the traditional definition for software reliability, that addresses software operating in a time-based manner according to specification for a period of time.

In practice, to scientifically certify software reliability for a software system, software usage testing must first be performed, that is to be presented in the next section.

Software Usage Testing

1. Software Usage Model

In principle, a software usage model, that plays a fundamental role in software usage testing [Walton95], characterizes various operational uses of a software system. An operational use is a skeleton for the intended use of the software in an intended environment. Thus, all possible operational uses of a software system constitute a population. A usage sample of test cases will be drawn statistically from the usage population. Performance on this sample is then used as a basis for the evaluation of software reliability.

Based on the functional specification and usage specification for the software, the built usage-model has many meaningful functions. It provides information not only to system developers concerning the importance (with probability of use) of each function, but for system testers to effectively construct test cases. Furthermore, the usage model even provides a framework for inferring software reliability through statistical testing.

2. Pocedures for Usage Tsting

In general, by the usage model approach, the method of software certification can be performed in the following steps [Chang96]:

(1) building a usage model which defines all possible events and their relationships,

(2) collecting the usage profile with probabilities for the different events,

(3) generating test cases statistically by the associated distribution,

(4) executing the test cases,

(5) collecting performance information and inter-failure data, and

(6) certifying the software by the reliability evaluation model.

As to the first step, various usage models have been proposed in literature, as summarized in Table 1.

Table1 Various software usage models

Model	Usage Item	Remarks	Reference
Domain based model	input values	cannot capture relations between inputs histories	[Hamlet92]
Algorithmic model	input values	inputs history is divided into classes, different usage profiles for different input history classes	[Woit94]
Operational profile	user mode, functions	use tree structure for different user categories cannot capture dynamic behavior.	[Musa93]
Grammar model	stochastic grammar	valid for some types of applications	[Dyer92]
State hierarchy model	user mode, services	valid for complex systems	[Wohin93]
Markov model	states, stimuli, transition probability	rich with theoretical derivation	[Whittaker94]

It is worth much attention to the Markov model provided on the last entry in the Table 1. In his study, Whittaker suggests that software testing is rather suitable to be treated as a stochastic process and, its usage be modeled by a finite state, discrete parameter, time homogeneous, irreducible Markov chain [Whittaker93]. Some successful projects have shown that the Markov approach to usage modeling has a tendency to make the testing process more efficient

[Whittaker94]. In this paper, Markov approach will be employed for usage modeling.

3. Issues on Usage Modeling

To perform the first step in the last section, Walton [Walton95] provides the following procedures in developing a Markov chain usage model:

(1) developing a model structure based on the specification of the software,
(2) assigning arc transition probabilities based on information available concerning intended use of the software and test objectives,
(3) computing usage statistics for the chain,
(4) reviewing the model and all usage statistics with respect to the software specification and the test plan, and
(5)revising the model as required to reflect realistic aspects of the application.

In general, the software usage model is initially developed from the software specifications. However, usually there is insufficient information to completely specify all one-step transition probabilities in the Markov chain, in particular for those newly developed systems. Considering such practical difficulty, three approaches to define transition probabilities have generally been used [Whittaker93]:

(1) The informed model, where transition probabilities are assigned based on field data.
(2) The intended model, where transition probabilities are assigned based on informed assumptions about expected use.
(3) The uniformed model, where, because no information is available concerning expected use of the software, uniform probabilities are assigned across the arcs for each state of the Markov chain.

Since the availability of information concerning expected use of the software is usually difficult to make thoroughly, a feasible and efficient practice by combining the above three approaches should be investigated. That is, initially individual transition probabilities may be assigned partially by some specific values that are fixed at the first time. Once the more detailed information was obtained

later, the same transition probability matrix may then be updated as real situation required.

Accordingly, an appropriate approach is then needed to generate and update dynamically a transition probability matrix for Markov chain usage models. In the following, this paper is trying to solve this problem by employing the quadratic programming approach.

Usage Modeling to Generate Transition Probabilities

1. Properties of a Markov Chain

Suppose that there exist n states in a usage model, and let $p_{i,j}$ denote the one-step stationary transition probability that the next state will be s_j given that the current state is s_i (i, j = 1 , ..., n). The Markov chain can then be represented as a two-dimensional, irreducible matrix **P** [Hillier90, p.563] as illustrated in the section E.

Furthermore, since a usage model is an irreducible Markov chain, the stationary distribution vector

$$\pi = \pi \mathbf{P} \qquad (2)$$

exists and is unique [Hillier90, p. 574]. The i-th element of π is the long run average probability of the occurrence of state s_i. To be specific, the stationary probability π_i is the probability of finding the chain in a certain state s_i after a large number of transitions, independent of the initial probability distribution defined over the states.

In terms of the software usage testing, π_i is the stationary probability of finding a Markov usage chain in a certain usage state s_i, independent of its initial transition probability matrix provided over the usage model.

The main advantage of using a Markov chain for usage modeling lies in that it allows software evaluators to perform significant, quantitative analysis that gives insight how the evaluation is likely to unfold. Some important results are listed as following[Walton95]:

(1) expected test case length, which can be used to help estimate resources required for testing,

(2) expected number of test cases required to cover all states in the usage model, which can be used to identify the amount of testing required to guarantee that each usage state was visited at least once in testing,

(3) expected proportion of time spent in each state in testing, which can be used to identify which usage functions will get the most attention from the test cases,

(4) expected number of test cases to first occurrence of each state, which can be used to identify states that are seldom visited.

Among the various criteria, minimization of the expected test case length for a Markov usage model will be studied for demonstrating the proposed approach.

2. Structural Constraints for a Markov Chain

For a Markov chain, in general, structural constraints that define whether a transition is allowed between two states may be represented by the following linear functions:

$$p_{i,j} = 0 \quad \text{for some i, j;} \tag{3}$$

$$p_{i,j} = 1 \quad \text{for some i, j; and} \tag{4}$$

$$\sum_{j} p_{i,j} = 1 \quad \text{for i} = 1, \ldots, n. \tag{5}$$

Besides, by the stationary properties,

$$\sum \pi_i = 1 \quad \text{for i} = 1, \ldots, n, \tag{6}$$

since each π_i is the corresponding resultant probability for each possible state in the usage model.

3. Formulation of Quadratic Programming

A Markov chain of n states represented by a transition matrix **P** will have n^2 transition probabilities. Of the n^2 probabilities, however, only those with values not equal to 0 or 1 need to be determined. Thus, on the case of usage modeling, the problem to be solved will become one of solving every decision variable for each transition probability not equal to 0 or 1.

As to considering the management policy, the objective function in this paper is defined to minimize the expected test case length, or the expected number of state transitions, for a certain state i, in a single test case denoted by $\mu_{i,i}$. However, an expected test case length for a usage state can be related to the reciprocal of its stationary probability measure as [Hillier90, p. 576]:

$$\text{Min } \mu_{n,n} = 1 / \pi_n , \tag{7}$$

where the state n is the terminate state of the chain.

Thus, the mathematical programming model for the investigated problem becomes, after considering the stationary properties:

$$\text{Max } \pi_n = \sum \pi_i p_{i,n} , \tag{8}$$

subject to the above structural constraints.

It is noted that the final form of this mathematical programming problem may be directly solved by a quadratic programming approach. That is, it may be immediately solved by the modified simplex method, through transforming first the objective function into a complementarity constraint with the Karush-Kuhn-Tucker (KKT) conditions [Hillier90, p. 523].

Illustration & Discussions

For demonstration of using the suggested approach, an example that was discussed in [Poore93] is shown as in Figure 1. After analyzing its software structure, the software system can be represented as a network of components. Each

directed arc, labeled with its transition probability, indicates that control passes from one component to another in the direction of the arrow. As discussed in the last section, availability of all one-step transition probabilities is usually not possible in the real situations. After field investigation of the given example, the transition probability matrix **P** may take the form as in Table 1, and with the following additional constraints:

$$p_{A,B} = 3\, p_{A,E} \tag{9}$$

$$p_{C,G} = 5\, p_{C,H} \tag{10}$$

$$\mu_{A,A} \leqq 7 \tag{11}$$

After employing the proposed method, the computed transition probabilities may be found in Table2. Note that the suggested algorithm also provides information that helps identifying components B, D and E as having highly critical function, which is straightforwardly obtained through sensitivity analysis under this quadratic programming process. During the software development phase, identification of some critical functions will guide the software design engineers to adapt the system to the more practical and reliable quantity.

This research proposes the quadratic programming approach to generating transition probabilities of a Markov usage model. The advantage of the suggested formulation lies that the fact there exist lots of appropriate computer packages available for use immediately. For instance, the LINDO 5.0 computer program [Schrage91] from Graduate School of Business, University of Chicago is implemented in this paper.

For comparison, Walton [Walton95] also solves the same problem, but she formulates the convex programming problem into an unconstrained Lagrangian one and then solves it by the Newton's algorithm. Although Walton's formulation may be solved by any unconstrained convex optimization solver, the proposed quadratic programming approach allows one to perform sensitivity analysis of model parameters. Furthermore, post-optimal analysis can also be easily carried out.

Conclusion

Software usage testing based on a usage model may provide a cost-effective use of the testing effort. Moreover, the testing result may support highly confident certification of software reliability measures in operational use by statistical inferences. In practice, usage modeling is the first task to perform the software usage testing, and is usually conducted in an iterative process. Any changes to the software specification or to the expected use of the software must be accompanied by the review each time and provided with appropriate modifications to the usage model.

This paper applies techniques from mathematical programming to present a new approach to generate and optimize the transition probability matrix. Essentially the proposed approach transforms the Markov usage chain into a quadratic programming model, which may be immediately solved by any mathematical programming package. The main advantage of the suggested methodology lies in that it may provide significant information of sensitivity analysis, which is valuable not only for improving software structure in software design process, but for adapting usage changes in usage modeling process.

In order to make use of software usage testing more efficiently and effectively, further research will continue in studying more techniques for improving usage modeling process.

References

1. Chang, W. K., *A Framework for Software Acceptance Process through the Statistical Usage Testing*, Technical Report THU-CIS-SEL-0021, Software Engineering Laboratory, Dept. of Computer & Information Sciences, Tunghai University, Dec., 1996.
2. Dyer, M., *The Cleanroom Approach to Quality Software Development*, John Wiley & Sons, 1992.
3. Hamlet, D., "Are We Testing for True Reliability?". IEEE Software, pp. 21-27, July 1992.

4. Hillier, F. S., Lieberman, G. J., *Introduction to Operations Research*, fifth edition, McGraw-Hill, 1990, p. 563.
5. Musa, J.D., "Operational profiles in Software-reliability Engineering," IEEE Software, March 1993, pp. 14-32.
6. Poore, J.H. and H.D. Mills and D. Mutchler, "Planning and Certifying Software System Reliability," IEEE Software, Jan. 1993, pp. 88-99.
7. Poore, J.H. and C.J. Trammell. *Cleanroom Software Engineering: A Reader*. Blackwell Publishers: Oxford, England, 1996.
8. Schrage, L., LINDO *User's Manual, Release 5.0*, the Scientific Press, San Francisco, U. S. A., 1991.
9. Walton, G. H., J. H. Poore and C. J. Trammell. "Statistical Testing of Software Based on a Usage Model," Software Practice and Experience, January 1995, vol. 25(1), pp. 97-108.
10. Walton, G.H., *Optimizing Software Usage Models*, Ph.D. Dissertation, Department of Computer Science, University of Tennessee, May 1995.
11. Whittaker, J. A. and J.H. Poore. "Markov Analysis of Software Specifications," ACM Transactions on Software Engineering and Methodology, January 1993, vol. 2(1), pp. 93-106.
12. Whittaker, J. A. and M. G. Thomason. "A Markov Chain Model for Statistical Software Testing," IEEE Transactions on Software Engineering, October 1994, 20 (10), pp. 812-824.
13. Wohin, C. and P. Runeson. "Certification of Software Components," IEEE Transactions on Software Engineering, June 1994, 20 (6), pp. 494-499.
14. Woit, D., "A Framework for Reliability Estimation," Proceedings IEEE 5th Int. Symp. on Software Reliability Engineering, Monterey, California, USA, pp. 18-24, 1994.

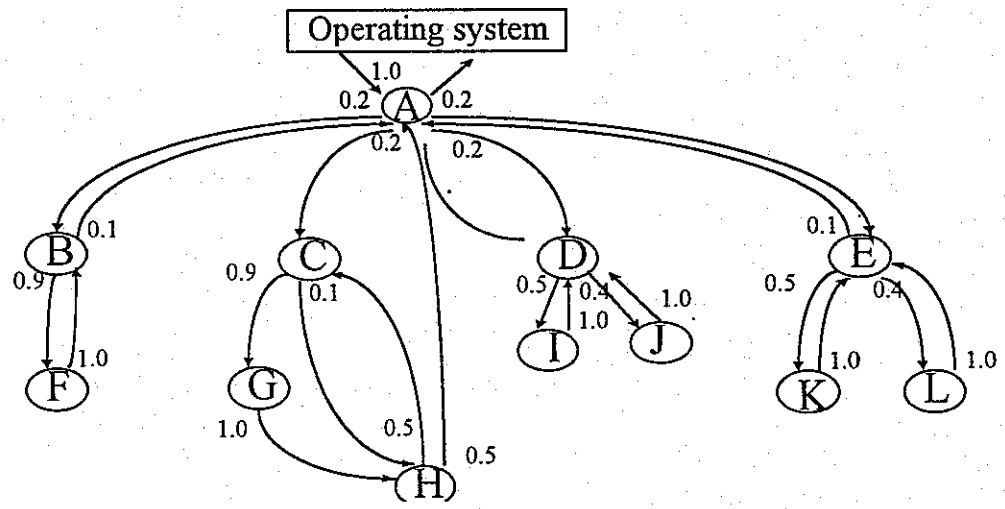

Figure1 Structure of a software system.

Table1 Considered transition probabilities for software structure in Figure 1.

State	OS	A	B	C	D	E	F	G	H	I	J	K	L
OS		1.											
A	0.2		?	?	?	?							
B	?						?						
C								?	?				
D		?								?	?		
E		?										?	?
F			1.										
G									1.				
H		?		?									
I					1.								
J					1.								
K						1.							
L						1.							

Table2 Computed transition probabilities for demonstration information.

State	OS	A	B	C	D	E	F	G	H	I	J	K	L
OS		1.											
A	0.2		0.45	0.05	0.15	0.15							
B	0.24						0.76						
C								0.83	0.167				
D		0.27								0.57	0.16		
E		0.33										0.44	0.21
F			1.										
G									1.				
H		0.72		0.28									
I					1.								
J					1.								
K						1.							
L						1.							

利用二次規劃法建構驗證軟體可靠度所需之使用模式

張文貴*

摘 要

在進行電腦軟體使用測試中，首先必須建立軟體使用模式，用來表示該軟體未來可能被使用的型態集合，而透過該軟體使用模式，則可進一步隨機產生一組測試案例的樣本，以便執行軟體使用測試時，估計軟體操作可靠度。

但是，在利用馬可夫鏈來表示軟體使用模式時，由於軟體未來可能被使用的不確定性，要指定所有的一次轉換機率，通常是相當困難的。在本文中，將先簡述軟體使用測試的方法及其特性，然後討論如何利用數學規劃法建立電腦軟體的使用模式，接著並以二次規劃法求解。利用此方法建立軟體使用模式，最主要的優點是，若某些模式參數後來更爲確定後，靈敏度分析之進行，將非常容易，可大量節省重新建立使用模式的工作時間。

關鍵詞: 軟體使用測試, 二次規劃法, 馬可夫鏈, 軟體可靠度驗證

*東海大學資訊科學系