

Volume 3, Issue 6, June 2013 ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

A Review of Markov Model for Estimating Software Reliability

Divya Bindal Department of CSE UIET kurukshetra university, Kurukshetra, Haryana, India

Abstract- Software reliability is the most important aspect of any software. The quality of a computer system tends to be greatly dependent on software, and is gaining in the importance of software reliability. Software Reliability is the ability of software to function under given environmental conditions for particular amount of time by taking into account all precisions of the software. Markov Model is used to represent the architecture of the software & provides a mean for analyzing the reliability of software. It is a mathematical system consisting of finite no of possible states that undergoes transition from one state to another state. The Markov Model assumes that future is independent of the past given the present i.e. the probability of transitioning from some state i to another state j depends only on the current state i not on the sequence of events that preceded it . In this paper we discussed how Markov Mode of software is drawn, its application and how it is used for estimating the reliability of software.

Keywords—Markov Model, Reliability, Failure Rate, Statistical Testing, Probability

I.

Introduction To Markov Model

The main function of most equipment and system depend more and more on software with the development of computer and information technology, but low reliability of software place a serious constraint on wide application of software, even lead to some disastrous result. Markov Model is used to represent the architecture of the software & provides a mean for analyzing the reliability of software "A Markov Model is a mathematical system that undergoes transitions from one state to another, between a finite or countable number of possible states. It is a random process usually characterized as memory less: the next state depends only on the current state and not on the sequence of events that preceded it" [1]. By using Markov Model a statistical model of software is drawn wherein each possible use of the software has an associated probability of occurrence. Test cases are drawn from the sample population of possible uses according to the sample distribution and run against the software under test. Various statistics of interest, such as the estimated failure rate and mean time to failure of the software are computed.

A sample three state Markov Model



Figure. 1 Markov Model

A usage model for a software system consists of states, i.e., externally visible modes of operation that must be maintained in order to predict the application of all system inputs, and state transitions that are labelled with system inputs and transition probabilities. To determine the state set, one must consider each input and the information necessary to apply that input. It may be that certain software modes cause an input to become more or less probable (or even illegal). Such a mode represents a state or set of states **in** the usage chain. Once the states are identified, we establish a start state, a terminate state , and draw a state transition diagram by considering the effect of each input from each of the identified states. The Markov chain is completely defined when transition probabilities are established that represent the best estimate of real usage.

Advantages of using Markov Model

- Simplistic Modeling Approach: Markov Models are simple to generate
- **Redundancy Management Techniques**: System reconfiguration required by failures is easily incorporated in the model.
- **Coverage:** Covered and uncovered failures of components are mutually exclusive events.
- **Complex Systems:** Many simplifying techniques exit which allow the modeling of complex systems.
- Sequenced Events: It allow computing of an event resulting from a sequence of sub events.

Disadvantage of Markov Model

The major drawback of Markov Model is the explosion of number of states as the size of system increases. The resulting models are large & complicated.

II. Types of Markov Model

The most common Markov models and their relationships are summarized in the following table [3]:

	System state is fully observable	System state is partially observable
System is autonomous	Markov Chain	Hidden Markov Model
System is controlled	Markov Decision Process	Partially observable Markov Decision Process

- Markov Chain: The simplest Markov model is the Markov chain. It models the state of a system with a random variable that changes through time. In this context, the Markov property suggests that the distribution for this variable depends only on the distribution of the previous state.
- **Hidden Markov Model:** A hidden Markov model is a Markov chain for which the state is only partially observable. In other words, observations are related to the state of the system, but they are typically insufficient to precisely determine the state.
- Markov Decision Process: A Markov decision process is a Markov chain in which state transitions depend on the current state and an action vector that is applied to the system. Typically, a Markov decision process is used to compute a policy of actions that will maximize some utility with respect to expected rewards.
- **Partially Observable Markov Decision Process:** A partially observable Markov decision process (POMDP) is a Markov decision process in which the state of the system is only partially observed. POMDPs are known to be NP complete, but recent approximation techniques have made them useful for a variety of applications, such as controlling simple agents or robots.

III. Software Reliability

Software Reliability is defined as the ability of software to function under given environmental conditions for particular amount of time by taking into account all precisions of the software [4].

Software reliability testing:

Software Reliability Testing is one of the testing field, which deals with checking the ability of software to function under given environmental conditions for particular amount of time by taking into account all precisions of the software. In Software Reliability Testing, the problems are discovered regarding the software design and functionality and the assurance is given that the system meets all requirements. Software Reliability is the probability that software will work properly in specified environment and for given time.

Probability = Number of cases when we find failure / Total number of cases under consideration

Using this formula, failure probability is calculated by testing a sample of all available input states. The set of all possible input states is called as input space. To find reliability of software, we need to find output space from given input space and software.

Objective of reliability testing

The main objective of the reliability testing is to test the performance of the software under given conditions without any type of corrective measure with known fixed procedures.

Secondary objectives

- 1. To find perceptual structure of repeating failures.
- 2. To find the number of failures occurring in specified amount of time.
- 3. To find the mean life of the software.
- 4. To know the main cause of failure.
- 5. After taking preventive actions checking the performance of different units of software.

Need of reliability testing

Software reliability is the most important aspect of any software. To improve the performance of software product and software development process through assessment of reliability is required. Reliability testing is of great use for software managers and software practitioners. Thus ultimately testing reliability of software is important.

Reliability metrics

Rate of Occurrence Of Failure (ROCOF) ROCOF measures the frequency of occurrence of failures .ROCOF measure of a software product can be obtained by observing the behaviour of a software product in operation over a specific time interval and then calculating the ROCOF value as the ratio of the total number of failures observed and the duration of observation.

Mean Time To Failure (MTTF) MTTF is the average time between two successive failures. To measure MTTF we can record the failure data for n failures. Let the failures occur at the time instants ti, t_{i+1}, t_2, \dots, t_n . Then MTTF can be calculated as

$$\sum_{i=1}^{n} \frac{t_{i+1} - t_i}{n-1}$$

Mean Time To Repair(MTTR) one failure occurs , some time is required to fix the error . MTTR measures the average time it takes to track the error causing the failure and to fix them.

Mean Time Between Failure(MTBF)The MTTF and MTTR metrics can be combined to get MTBF metric:

MTBF=MTTF+MTTR.

Probability Of Failure On Demand POFOD is the likelihood of the system failing when a service request is made.

Availability Availability of a system is a measure of how likely would the system be available for use over a given period of time

Availability=
$$\frac{MTTF}{MTTF+MTTR}$$
*100%

IV. TESTING PROCESS

The typical process followed when performing statistical testing is presented in Figure 3.

- Get Specification: Some form of a specification detailing the correct behaviour of the software is needed to develop the usage model. The correct behaviour of the software may be defined by a formal specification, requirements documentation, user's manual, or a predecessor system.
- **Develop Model Structure:** The states of use and arcs connecting the states are identified. In current practice this stage is manual, i.e., the structure of the usage model cannot be derived automatically from a specification or other artefact. However, work is in progress to show how to derive the model structure from a sequence-based specification of the software

Testing Process



Figure3 Testing Process

- Assign Probabilities: The state transition probabilities of the usage model are assigned. The probabilities may be assigned manually or they may be represented as a set of constraints and automatically calculated to satisfy some testing goal.
- Verify and Analyze Model: Analytical results are calculated to aid in test planning and verifying that the model properly represents the expected use of the software
- Run Non-Random Tests: Non-random test cases are crafted or generated from the usage model and then executed on the software under test. Examples of non-random tests currently being used are hand crafted tests, test cases generated in order of probability of occurrence, and test cases generated to cover all arcs in the usage model in the minimum number of testing steps. Non-random tests may be run to satisfy contractual obligations, explore a particular use of the software, help validate the usage model and test facility, or determine whether the software is stable enough for full scale testing. Current practice does not include use of non-random tests in reliability estimation.
- Estimate Reliability: The testing record containing information as to which tests were run and where failures were observed in the test cases is used to estimate the reliability of the software.
- **Decide Whether to Stop Testing**: The testing record is evaluated to determine whether testing should continue or stop.
- **Stop and Report Results:** After testing is finished the test results may be used for a number of purposes, such as deciding whether to release the product, evaluating whether the software development process is under control, or evaluating the performance of a new piece of technology used in the product.

V. WORK DONE

Ambuj Goyal *et al.* [5] presented a unified framework for simulating Markovian models of highly dependable systems. They used a variance reduction technique called Importance Sampling to speed up the simulation by many orders of magnitude over standard simulation. This technique was combined very effectively with regenerative simulation to estimate measures such as steady state availability and mean time to failure. Moreover, it was combined with conditional Monte Carlo methods to quickly estimate transient measures such as reliability, expected interval availability, and the distribution of interval availability. This paper shown that importance sampling may be fruitfully applied in conjunction with a variance reduction method known as conditioning which made the stochastic behaviour of the time spent in the fully operational state was easy to calculate.

James A. Whittaker *et al.* [6] described in detail the usage analysis and inference procedure including various computations on the ensuing Markov chains. Stopping criteria were developed and a discrete software reliability model was presented. The certification for the Cleanroom methodology was performed by used a stochastic model. Software usage was modelled by a finite state Markov chain. The Markov Chain was used as a test case generator and computed analytical descriptions of the sequences. These computations were allowed for a thorough analysis of expected usage patterns. A second Markov chain was initialized and allowed to evolve according to the results of the test cases those were applied on the software. This second chain was used to compute stopping criteria based upon the testing process reached to a steady state & also described how it was used for computing the reliability and the MTBF of the software.

James A. Whittaker *et al.* [7] described a method for statistical testing based on a Markov chain model of software usage. Firstly it described the construction of a Markov chain as a sequence generator for statistical testing & shown how analytical results associated with Markov chains can aid in test planning. Secondly it described the construction of second order Markov chain that encapsulated the history of the test & included any observed failure information. The influence of the failures was assessed through analytical computations on that chain. Then derived a stopping criterion for the testing process based on a comparison of the sequence generating properties of the two chains. The analysis of the testing chain was done in this paper was intended as a supplement to the many reliability models that existed in the literature. The testing chain discussed in this paper represented a new perspective on test data.

Alberto Avritzer *et al.* [8] presented three algorithms for automatically generating test suites to test the resource allocation behaviour of software systems that were modelled by Markov chains. A tool was built that does the automatic test suite generation and used it to generate suites for five large industrial software telecommunication systems. This paper also introduced a new approach for software reliability that was used to assess software that has been tested by using load testing algorithms, and applied that to assess the reliability of the five industrial systems. The initial results were extremely encouraging, with projects reporting the detection of serious program faults that would not have been detected until field release, had these algorithms not been used but the reliability notion described in that paper were used to guide the user in the assessment of the severity of observed failures.

Katerina Goseva-Popstojanova *et al.* [9] proposed the software reliability modelling framework that considered the phenomena of failure correlation and studied its effects on the software reliability measures. The important property of the developed Markov renewal modelling approach was its flexibility. It was allowed the construction of the software reliability model in both discrete time and continuous time, and the analysis was based on Markov chain theory or Thus, the modelled approach was an important step toward more consistent and realistic modelling of software reliability.

It was related to existing software reliability growth models. Many input-domain and time-domain models can be derived as special cases under the assumption of failure s-independence. It was shown that the classical software reliability theory can be extended to consider a sequence of possibly s-dependent software runs, via, failure correlation.

Chaitanya Kallepalli *et al.* [10] developed an approach for statistical Web testing and reliability analysis supported by automated information extraction from existing Web logs. The general model used was the Unified Markov Model. Existing tools and newly developed utility programs were used to support the construction of UMM and the related reliability analyses. Existing tools were used to extract usage information. The additional effort was done only for the implementation of proposed approach to a reasonable level. The combination of existing tools and newly developed utility programs gave a comprehensive yet inexpensive alternative to the automated support of the proposed approach. This approach has been applied to analyze the log files of the Web pages & a study was also carried out to evaluate the hypothetical effectiveness of their statistical Web testing strategy.

Jayant Rajgopal *et al.* [11] proposed a procedure that used Markovian model for assessing the reliability of software that was decomposed into a finite no of modules. Markovian model was used for the transfer of control between modules in order to develop the system reliability expression in terms of module reliabilities. An operational test procedure was considered in which only the individual modules were tested & system was considered acceptable if, and only if, no failures were observed, the minimum no of test required of each module was determined such that probability of accepting a system whose reliability falls below a specified value R_0 was less than a specified small fraction β . This sample size determination problem was formulated as a two-stage mathematical program and an algorithm was developed for solving that problem. Author observed cost of testing were diff from one module to another by used this changed the function of linear program which would now be $\sum_i c_i k_i$ instead of $\sum_i k_i$ where c_i was the test cost for module i.

Shereef Abu Al-Maati *et al.* [12] compared dynamic allocation model that dynamically refines allocation of test cases during the reliability testing process with the theoretical optimal model through Monte Carlo simulation. In this paper authors focused on the partition testing strategy which is composed of two phases:

- 1. A partitioning phase (The partitioning phase focused on dividing the programs input domain) and
- 2. A test case allocation phase(The test case allocation phase focused on how to distribute the test cases to each partition in such a way as to improve the estimated reliability of the software)

The simulation results presented in this paper indicated that the two stage allocation model performed optimally.

Katerina Goseva–Popstojanova *et al.* [13] proposed a methodology for uncertainty analysis in architecture–based software reliability models that were suitable for large complex component based applications and applicable throughout the software life cycle. Firstly authors described different approaches to build the architecture based software reliability model and to estimate parameters. Then, they performed uncertainty analysis using the method of moments and Monte Carlo simulation which enabled to study how the uncertainty of parameters propagates in the reliability estimate. Both methods were illustrated on two case studies and compared using several criteria. The uncertainty analysis provided richer measures of software reliability than the traditional point estimate. These measures was used for guiding allocation of testing efforts, made quantitative claims about the quality of the software subjected to different operational usages, and for reliability certification of component–based software systems.

Fenhua Zhen *et al.* [14] described system test methodology based on the Markov Chain Usage Model for this purpose they presented a framework for transforming the Unified Modelling Language-Sequence Diagram (UML-SD) to the Markov chain whose encoding rule was a kind of Markup language (Markov Chain Markup Language-MCML). Then, the statistical software testing based project-Markov Test Logic (MaTeLo), together with those technologies that used in MaTeLo was introduced in this paper. The algorithm for translating UML-SD into Markov chain usage model (MCUM) and its corresponding tool implementation was denoted in this paper. The algorithm presented in this paper provide a graphic view of the MCUM & this algo made the generation of test cases based on the MCML files quite straight forward.

Jiong Yan *et al.* [15] presented a method to generate the usage model from real-time software UML models. Firstly authors defined the reasonably constrained real-time software UML artefacts, which included use case diagrams, timed sequence diagrams and the execution probability of each sequence diagram in its use case. Secondly author presented a method that derived the software usage model from the constrained UML artefacts. The distinguished feature was the incorporation of timing constraints in the sequence diagrams .With the additional descriptions of timing constraints and statistical testing constraints in UML models, this method can generated the corresponding Markov chain usage models of the software systems, and facilitated real-time software statistical testing.

Helene Le Guen *et al.* [16] this paper presented an improved reliability estimation and coverage measure for SUT when Markov chains were used. These measures were implemented in a new tool .Markov chain was used here to represent, as an automaton of the dynamical behaviour of the software. Markov chain is also used to represent the architecture of the software. The reliability approach presented in this paper was employed in two situations. The two situations were the solution proposed by Whittaker's and the solution proposed by Sayre's.

The new measures were implemented together with Whittaker's and Sayre's approach in a tool (named MaTeLo). The Improvements of the implemented tool were:

- 1. Test cases were randomly generated from a profile.
- 2. The reliability estimation reflected how much of the usage model was covered.
- 3. The test cases do not reveal a failure; the reliability was not equal to 1.

Swapna S. Gokhale *et al.* [17] proposed a unifying framework for state-based models applied to architecture-based software reliability prediction. State-based models used the control graph to represent software architecture, and predicted reliability analytically. To predict the reliability of software they outlined the information necessary for the specification of the state based models of the software & also proposed a systematic classification scheme for state-based approaches to reliability prediction. The classification scheme was considered three aspects while categorizing the models:

- 1) The model used to represent the architecture of the application,
- 2) The model used to represent the failure behaviour of its components, and
- 3) The solution method.

They placed the existing state-based models in the literature in their appropriate categories according to the above three aspects & then present an exhaustive analysis of those state-based models where the architecture of the application was modelled either as a discrete time Markov chain (DTMC), or a continuous time Markov chain (CTMC). A detailed discussion regarding the input parameters required by each model, and how these parameters may be estimated from the different software artefacts were provided. Depending on the software artefacts available during a given phase of the software life cycle, and the parameters estimated from these artefacts, they provide guidance regarding which model may be appropriate to predict the reliability of an application during each phase of the software life cycle.

Leslie Cheung *et al.* [18] identified several sources of uncertainties, and illustrated how to incorporate them into reliability modelling framework. The presence of uncertainties, due to the lack of information about a software system, was a major challenge to any architectural-level reliability modelling technique. They tried to solve that problem & then discussed and evaluate how well their component reliability prediction framework proposed in addresses these uncertainties.

Shiyi Xu [19] proposed a new approach called orderly random testing using PDDTS(Predetermined Test Sequence), attempted to made the Semi-Anti-Random Testing Sequence more effective .Random testing was seemed to be inefficient for its random selection of test patterns. Therefore, a new concept of pre-determined distance among test vectors was proposed in this paper to make it more effective in testing. The idea was based on the fact that the larger the distance between two adjacent test vectors in a test sequence, the more the faults were detected by the test vectors. Procedure of constructing such a testing sequence was presented in detail. Experimental evaluating results on hardware circuits have essentially justified the methods presented here. It was also shown that this method could also be efficiently employed in software testing, especially for a large scale of software system but some of theoretical proofs still need to be solicited.

Yi WAN *et al.* [20] proposed Software reliability analysis Markov model by Markov theory and mathematical statistics principle based on repairable debugging characteristic of software system. If all units that were presented in software repaired to new normal state after fault called reparable debugging characteristics of software in that case system behaviour was able to be described by homogeneous Markov Chain. Software reliability was analyzed by Markov theory. Effective measures based on Markov model were put forward, It was important to reduce software fault and improve operational quality of software. The mathematical statistics principle was combined with Laplace transformation as a result software reliability synthetically analysis model was built and evaluation characteristic parameters were obtained. Availability, reliability and mean time-to-failure were derived in that paper, many results were obtained by actual example analysis. It provided a theoretical basis and new method for reliability design and scientific management of software.

Katerina Goseva-Popstojanova *et al.* [21] addressed the problem of estimating software reliability when the successive software runs were statistically correlated. First, they generalized their previous work by used higher order Markov chain to model a sequence of dependent software runs. Then they conducted an empirical study for exploring the phenomenon of dependent software runs using three software applications as case studies. Based on two statistical approaches "One based on the difference between the observed and expected frequencies of sequences and the other based on the information theory", They showed that the outcomes of software runs (i.e., success or failure) for two of the case studies were dependent on the outcome of one or more previous runs, in that cases first or higher order Markov chain models were appropriate. Finally, they estimated the parameters of the appropriate models and discussed their effects of dependent software runs on the estimates of the software reliability.

Jianwen Chen *et al.* [22] proposed Bounded Monte Carlo Method. It was an improvement upon the basic Monte Carlo Method which can increase the computing accuracy with the help of lower and upper bounds. It was applied to the cases where the lower and upper bounds can be estimated easily before the basic Monte Carlo Method was invoked.

This paper illustrated it in the two examples - computing the number π and computing event probability. These examples were shown that a more accurate result can be obtained with the help of lower and upper bounds. In the Bounded Monte Carlo Method, the basic Monte Carlo Method was repeated many times that increased the cost and time.

Yashodhan Kanoria et al. [23] presented a new technique for statistical static timing analysis (SSTA) based on Markov chain Monte Carlo (MCMC), that allowed fast and accurate estimation of the right-hand tail of the delay distribution.



Figure 4.1

A "naive" MCMC approach was inadequate for SSTA. Several modifications and enhancements were presented in this paper that enabled application of MCMC to SSTA.

The new technique was also used for:

- 1. Yield calculation
- 2. Critical path identification
- Shipped-product Quality Loss (SPQL) estimation 3.

ASIberto Avritzer et al. [24] proposed an automatic test case generation approach by applied performability theory. Performability theory said that the requirements were specified in terms of a chosen performance metric. The expected response time and the number of tasks executed per time unit (throughput) were the two common metrics. Both the reliability modelling and test case generation approaches that were presented in this paper used the function p(n, t), the transient value of the probability of program P's correct execution, for input "n" and time "t". Two case studies were presented to illustrate their proposed approach. Proposed model-based test case generation approach was used to assess the reliability of large industrial mission-critical systems

Thanh-Trung Pham et al. [25] presented an approach to predict the reliability of component-based systems. Reliability prediction methods for component-based systems used Markov models were often limited to a model of sequential executions. This approach relaxes these constraints by incorporating error propagation analysis and multiple execution models together consistently. It helps to improve the quality of the system in a cost-effective manner.

Bo Zhou et al. [26] proposed a software random testing scheme based on Markov chain Monte Carlo (MCMC) method for addressing the problem associated with random testing. Random testing (RT) was one of the most classical software testing strategies. Although the RT was quite simple for implementation, it was often argued that RT was inefficient. A probability model was proposed to represent the activities for finding failures in software testing. In experiments, author compared effectiveness of MCMC random testing with both ordinary random testing and adaptive random testing in real program sources. These results provided that MCMC random testing was drastically improved the effectiveness of software testing.

VI. Conclusion and Future Scope:

Software is an essential component of many safety critical systems. These systems depend on the reliable operation of software components. Software reliability is the probability of fault free operation of software components in a specified period of time in a specified environment. A number of studies have been conducted for measuring reliability of given software as a result a number of analytical models has been introduced. One of them is the Markov Model that utilizes Markov Property & makes use of software architecture which is defined as the set of components, connectors & configurations i.e. it is an architecture based software reliability model. This paper discussed how Markov Model of software is drawn, its application & how it is used for estimating the reliability of software. Future work is:

- Step1 Markov Modeling for web application.
- Step2 Test case construction.
- Step3 Reliability assessment and predictions on test result.

Acknowledgement

I would like to express my special thanks and gratitude to my guide Mr. Karambir who gave me the golden opportunity to write this paper on the topic A Review of Markov Model for Estimating Software Reliability, which also helped me in carrying out an intensive research through which I came across so many new concepts and terminologies. I am really thankful to him.

References

- [1] MS Swaminathan, "*Markov Models and Related Procedures*", SCOPE 34 Practitioner's Handbook on the Modelling of Dynamic Change.
- [2] Norman B. Fuqua "*The Applicability of Markov Analysis Methods to Reliability, Maintainability & Safety*" START Volume10, Number2,2003.
- [3] Leslie Pack Kaelbling; Michael L Littman & Anthony R Cassandra, "*Planning and acting in partiall observable stochastics domains*," Artificial Intelligence (Elsevier). Volume 101, Issues 1–2: 99–134,1998
- [4] Roger Pressman, "Software Engineering A practitioner Approach", Mc.GrawHill.
- [5] Ambuj Goyal, Penvez Shahabuddin, Philip Heidelberger, Victor F. Nicola and Peter W. Glynn "A Unified Framework for Simulating Markovian Models of Highly Dependable Systems" IEEE Transactions On Computers, VOL. 41, NO. 1, January 1992.
- [6] James A. Whittaker and J. H. Poore "Statistical Testing for Cleanroom Software Engineering" in 1992 IEEE.
- [7] James A. Whittaker and Michael G. Thomason "*A Markov Chain Model for Statistical Software Testing*" IEEE Transactions on software engineering VOL. 20, NO. IO, October 1994.
- [8] Alberto Avritzer and Elaine J. Weyuker "*The Automatic Generation of Load Test Suites and the Assessment of the Resulting Software*" IEEE Transactions On Software Engineering, VOL. **21**, NO. 9, September 1995.
- [9] Katerina Goseva-Popstojanova & Kishor S. Trivedi "*Failure Correlation in Software Reliability Models*" IEEE Transactions on reliability , VOL. 49, NO. 1, MARCH 2000.
- [10] Chaitanya Kallepalli & Jeff Tian "*Measuring and Modeling Usage and Reliability for Statistical Web Testing*" IEEE Transactions on software engineering VOL. 27, NO. 11, November 2001.
- [11] Jayant Rajgopal and Mainak Mazumdar "*Modular Operational Test Plan for Inferences on Software Reliability Based on a Markov Model*" IEEE Transactions On Software Engineering, VOL. **28**, NO. 4, April 2002.
- [12] Shereef Abu Al-Maati & Kamel Rekab "Dynamic Test Allocation Model for Software Reliability" in proceedings of the Third International Conference On Quality Software (QSIC'03) 2003 IEEE.
- [13] Katerina Goseva–Popstojanova and Sunil Kamavaram "Assessing Uncertainty in Reliability of Component– Based Software Systems" Proceedings of the 14th International Symposium on Software Reliability Engineering (ISSRE'03) 2003 IEEE.
- [14] Fenhua Zhen and Chenglian Peng "A System Test Methodology Based on the Markov Chain Usage Model" The 8th International Conference on Computer Supported Cooperative Work in Design Proceedings 2003.
- [15] Jiong Yan, Ji Wang and Huo-wang Chen "Automatic Generation of Markov Chain Usage models from Realtime Software UML Models" Proceedings of the Fourth International Conference on Quality Software (QSIC'04) 2004 IEEE.
- [16] Helene Le Guen, Raymond Marie and Thomas Thelin "*Reliability Estimation for Statistical Usage Testing using Markov Chains*" Proceedings of the 15th International Symposium on Software Reliability Engineering (ISSRE'04) 2004 IEEE.
- [17] Swapna S. Gokhale, Kishor S. Trivedi "Analytical Models for Architecture-Based Software Reliability Prediction: A Unification Framework" IEEE Transactions On Reliability, VOL. 55, NO. 4, December 2006.
- [18] Leslie Cheung, Leana Golubchik, Nenad Medvidovic, Gaurav Sukhatme "Identifying and Addressing Uncertainty in Architecture-Level Software Reliability Modeling" in 2007 IEEE.
- [19] Tomohiko Takagi and Zengo Furukawa "Construction Method of a High-Order Markov Chain Usage Model" 14th Asia-Pacific Software Engineering Conference 2007.
- [20] Yi Wan, Chengwen WU "Software reliability model based on stochastic theory" in 2009 IEEE.
- [21] James A. Whittaker and Michael G. Thomason "*A Markov Chain Model for Statistical Software Testing*" IEEE Transactions on software engineering VOL. 20, NO. IO, October 1994.
- [22] Jianwen Chen and Ling Feng "Using Lower and Upper Bounds to Increase the Computing Accuracy of Monte Carlo Method" International Conference on Computational and Information Sciences 2010.
- [23] Yashodhan Kanoria, Subhasish Mitra and Andrea Montanari "Statistical Static Timing Analysis uses Markov Chain Monte Carlo" in 2010 EDAA.
- [24] A. Avritzer, E. de Souza e Silva, R.M.M. Leao and E.J. Weyuker "Automated generation of test cases using a *performability model*" journal Published in IET Software Received on 29th March 2011.
- [25] Thanh-Trung Pham, Xavier Defago "Reliability Prediction for Component-based Systems: Incorporating Error Propagation Analysis and Different Execution Models" 12th International Conference on Quality Software 2012.
- [26] Bo Zhou, Hiroyuki Okamura and Tadashi Dohi "Enhancing Performance of Random Testing through Markov Chain Monte Carlo Methods" IEEE Transactions on Computers, VOL. 62, NO. 1, January 2013.