Global Journal of Computer Science and Technology
Volume 11 Issue 8 Version 1.0 May 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
ISSN: 0975-4172 & Print ISSN: 0975-4350

Software Reliability Simulation: Process, Approaches and Methodology

By Javaid Iqbal, Dr. S.M.K. Quadri
University of Kashmir

Abstract- Reliability is probably the most crucial factor to put ones hand up for in any engineering process. Quantitatively, reliability gives a measure (quantity) of quality, and the quantity can be properly engineered using appropriate reliability engineering process. Software Reliability Modeling has been one of the much-attracted research domains in Software Reliability Engineering, to estimate the current state as well as predict the future state of the software system reliability.
This paper aims to raise awareness about the usefulness and importance of simulation in support of software reliability modeling and engineering. Simulation can be applied in many critical and touchy areas and enables one to address issues before they these issues become problems. This paper brings to fore some key concepts in simulation-based software reliability modeling. This paper suggests that the software engineering community could exploit simulation to much greater advantage which include cutting down on software development costs, improving reliability, narrowing down the gestation period of software development, fore-seeing the software development process and the software product itself and so on.

Keywords: Software Reliability Engineering, Software Reliability, Modeling, Simulation, Simulation model.

GJCST Classification: C.4, D.2.4

SOFTWARE RELIABILITY SIMULATION PROCESS, APPROACHES AND METHODOLOGY

Strictly as per the compliance and regulations of:

© 2011 Javaid Iqbal, Dr. S.M.K. Quadri. This is a research/review paper, distributed under the terms of the Creative Commons Attribution-Noncommercial 3.0 Unported License http://creativecommons.org/licenses/by-nc/3.0/), permitting all non-commercial use, distribution, and reproduction inany medium, provided the original work is properly cited.

Software Reliability Simulation: Process, Approaches and Methodology

Javaid Iqbal[α], Dr. S.M.K. Quadri[Ω]

Abstract- Reliability is probably the most crucial factor to put ones hand up for in any engineering process. Quantitatively, reliability gives a measure (quantity) of quality, and the quantity can be properly engineered using appropriate reliability engineering process. Software Reliability Modeling has been one of the much-attracted research domains in Software Reliability Engineering, to estimate the current state as well as predict the future state of the software system reliability.

This paper aims to raise awareness about the usefulness and importance of simulation in support of software reliability modeling and engineering. Simulation can be applied in many critical and touchy areas and enables one to address issues before they these issues become problems. This paper brings to fore some key concepts in simulation-based software reliability modeling. This paper suggests that the software engineering community could exploit simulation to much greater advantage which include cutting down on software development costs, improving reliability, narrowing down the gestation period of software development, fore-seeing the software development process and the software product itself and so on.

Keywords- Software Reliability Engineering, Software Reliability, Modeling, Simulation, Simulation model.

I. Introduction

Owing to the unexpectedly spiraling increase in the size and complexity of software systems during the past few decades, software reliability has become even more increasingly important for such massive systems. As a result of the compound growth rate of the order of ten times every five years in the size and complexity of software systems deployed in the key areas of telecommunications, defense, transportation industries, business etc, software system reliability is the prime factor to check out for. In such systems, a software failure can lead to serious, even fatal, consequences and repercussions in safety-critical and mission-critical systems as well as in normal business.

Software system reliability stands out as the key benchmark attribute for a software system among its various attributes. The levels of service dependability of a software system during its life-time are the indications for its reliability. In fact, the performance criterion of a software system is known by how long the software system will render faithful service. As a result of spiraling increase in the complexity of software systems, performance analysis of the software systems has gained further attention. Much focus has gone to the structural side of software systems as well. In general, the various components of a software system must remain expectedly faithful vis-à-vis their intended functions and deliverables. Software reliability has been dominating the thought-process ever since the size and hence complexities of software systems have increased. As fallout of increased size and complexity of software systems, factors contributing to the unreliability of the system become more pronounced. However, even though some level of unreliability does exist for a software system, it is worthwhile to express the quality of the software system by measuring some objective attributes such as reliability and availability. Software reliability characterizing the dynamic quality attribute of a software system can measure and predict the operational/usage profile of the software system.

About[α]- Assistant Professor in the P.G Department of computer Science, University of Kashmir- North Campus (India). (Telephone: (91)9596032499 email: pzjavaidz@yahoo.co.in)

About[Ω]- Head P.G Department of computer Science, University of Kashmir–Hazratbal Campus(India). (Telephone: (91)9419426555 email: quadrismk@hotmail.com)

II. Software Reliability and Software Reliability Engineering

Software Reliability is defined as the probability that software will provide failure-free operation in a fixed environment for a fixed interval of time [17]. In fact, software reliability is the key attribute in software reliability engineering which stands out among other attributes of software quality such as functionality, usability, capability, maintainability, and, etc., for its relevance to quantifying software failures. Software reliability quantifies software failures in a software system. By definition Reliability is probabilistic and hence hard to quantify accurately.

Software Reliability Modeling has been an active research domain for fault/failure forecasting, in software reliability engineering, for estimation as well as prediction of the current and future states, respectively, of the reliability of a software system. A software reliability model represents the behavior of software failures with respect to time as a random process. Reliability modeling as an essential element of the reliability estimation process determines whether a software system meets the specified levels of reliability and thus can be used to decide about the release time of a software system. Software Reliability Engineering (SRE) encompasses certain engineering techniques for the development and maintenance of software systems with an objective of measuring and predicting reliability (quality) as a quantity. The estimation as well as the prediction of reliability of a software system, involves the use of failure data represented as failure process through its reliability model. Probabilistic approach, being most common approach to developing software reliability models, represents the failure occurrences and the fault removals as probabilistic events. Probabilistic software reliability models are classified into various classes, including error seeding models, failure rate models, curve fitting models, reliability growth models, Markov structure models, and non-homogeneous Poisson process (NHPP) models .The three main reliability modeling approaches are: the error seeding and tagging approach, the data domain approach, and the time domain approach. Among these the time domain approach has gained much acceptance where techniques like curve-fitting and extrapolation are used. However, SRE techniques do have their limitations too.

Limitations of some existing SRE techniques:

1) Late collection of failure data: - SRE techniques collect the failure data during integration testing or system testing phases thereby providing for little flexibility in design re-considerations and re-structuring.

2) Non-Exhaustive failure data: - failure data collected by testing does not cover all failures under all settings (environmental, operational, usage etc). The side-effects are especially visible in software systems where we need to maintain highest levels of reliability. As a result, reliability estimation and prediction using the restricted testing data may only be approximations.

3) Non-realistic assumptions: - Assumptions underlying various SRE techniques or modeling methods for the reliability estimation may be too much unrealistic and optimistic in relation with the real scenario pertaining to the problem at hand.

Moreover, software testers test software as per laboratory settings without referring to its environmental settings. In an effort to break the software much of their attention and effort goes to designing of test cases for exceptional and boundary conditions, rather than testing for normal routine operations. Software reliability measurers, on the other hand, are much focused on testing of software as per its operational profile in order to allow for accurate reliability estimation and prediction. Against this backdrop of limitations of SR techniques and mutually exclusive focuses of testers and reliability measurers, simulation offers a luring approach to reliability modeling of a software system for it has the scope to address these important issues and bottlenecks. Furthermore, effective reliability modeling ultimately requires good data sets which are faithfully comprehensive, complete, or consistent. Such data sets are very rarely collected owing to different factors. However, simulation-based approaches do hold promise for such scenarios as well.

III. Simulation

a) *General Description*

Simulation is experimentation with models. More specifically, simulation is the technique of imitating the character of an object or process in such a way that enables us to make quantifiable inferences about the real object or process being simulated [25],[26]. When simulation is applied to software reliability, it can be used to mimic key characteristics of the various processes involved. To study a system, it is possible to experiment with the system itself or with the model of the system; experimenting with the system itself may be not be viable and feasible always, depending on the nature and type of system to be studied. Cost and risk analysis may not permit it. The objective, however, is to comprehend and predict how a system will perform before it is built. Consequently, the study of the system under consideration is generally conducted with a model of the system. A model is not only a substitute of a system but also a simplification of the system.

A model is an abstraction (simplification) of a real (existent) or conceptual (non-existent) system that is itself complex. A model embodies only those characteristics of the system under study, which are required for study, prediction, modification, or control of the system. Thus, a model includes some, but not all, aspects of the system being modeled. A model provides useful insights, predictions, and answers to the questions it is used to address. However, no single model can be used in all the situations. No model is complete; one model may work well for a set of certain software, but may be completely off track for other kinds of problems [1].

Increasing familiarization of the object-oriented systems design and modeling as well as the strikingly impressive web-based system developments have led to a substantial rise in the use of component-based software development approaches [10]. With the availability of commercial off-the-shelf software components (COTS), development of in-house, or out-sourced components, the whole application development takes place under a heterogeneous environment (multiple teams in different environments) and hence it may be inappropriate to describe the overall reliability assessment of such applications using only one of the several software reliability growth models- the black-box approach[6]. Thus, for such structured and component-based software systems, reliability prediction must start as early as its architecture phase of the life-cycle. Moreover, assessment of reliability is made in terms of reliabilities of the components of such systems.

The existing analytical methods to predict the reliability of component-based systems are based on the Markovian assumption [2], [13]. Semi-Markov [14] relaxes this assumption in a restrictive manner. However, both Markovian and semi-Markovian methods to predict the reliability of such heterogeneous systems suffer from several limitations:

They are subject to an intractably large state-space, and They cannot account for the influence of various parameters such as reliability growth of the individual components, dependencies among the components, etc., in a single model.

Nevertheless, methods are in place to model the reliability growth of the components which cannot be accounted for by the conventional analytical methods [8], [9], [15], but they are again subject to the state-space explosion problem, and their computational complexity is also a problem. There are many other methods, however, there is no single analytical model, which takes into account all such features, and is not intractable.

A simulation model, on the other hand, offers an attractive alternative to analytical models as it describes a system being characterized in terms of its artifacts, events, interrelationships and interactions in such a way that one may perform experiments on the model, rather than on the system itself, ideally with indistinguishable results [1][26]. Thus a simulation model can capture a detailed system structure, and facilitate the study of the influence of various factors such as reliability growth, repair policies, correlations among the various versions etc. Simulation can also represent the impact of several strategies that may be employed during testing and different deployment configurations during operation. Simulation can be used to study the influence of different factors separately as well as in a combined manner on dependability measures. In addition, simulation techniques can be provided for SRE purposes. They can produce observables of interest in reliability engineering, including discrete integer-valued quantities that occur as time progresses.

b) *Simulation Model*

A simulation model is a computerized model that represents some dynamic system or phenomenon and possesses the characteristics of interest of study about that system. A simulation model or any other modeling method is an inexpensive way to gain important insights when the costs, risks, or logistics of manipulating the real system of interest are prohibitive. The most common purposes of simulation models are to provide a basis for experimentation, predict behavior, answer "what if" questions, teach about the system being modeled, etc. A simulation model used in a simulation study is basically a parametric model where the values of the parameters need not be specified. It consisting of a particular parameter set which represents the values of the parameters of the model.

Static models or techniques may not always be the first choice for a system with high complexity levels. In such systems, simulations are generally employed to model the complexity of the system which may manifest itself in the form of system uncertainty and stochasticity, dynamic system behavior and feed-back and feed-forward mechanisms. For uncertain systems simulation provides a flexible and useful mechanism for capturing uncertainty related to complex systems. For systems with dynamic behavior, dynamic simulation models are very flexible and support modeling of a wide variety of system structures and dynamic interactions. For systems with feedback mechanisms where behavior and decisions made at one point in the process impact others in complex or indirect ways, simulation is a usable alternative.

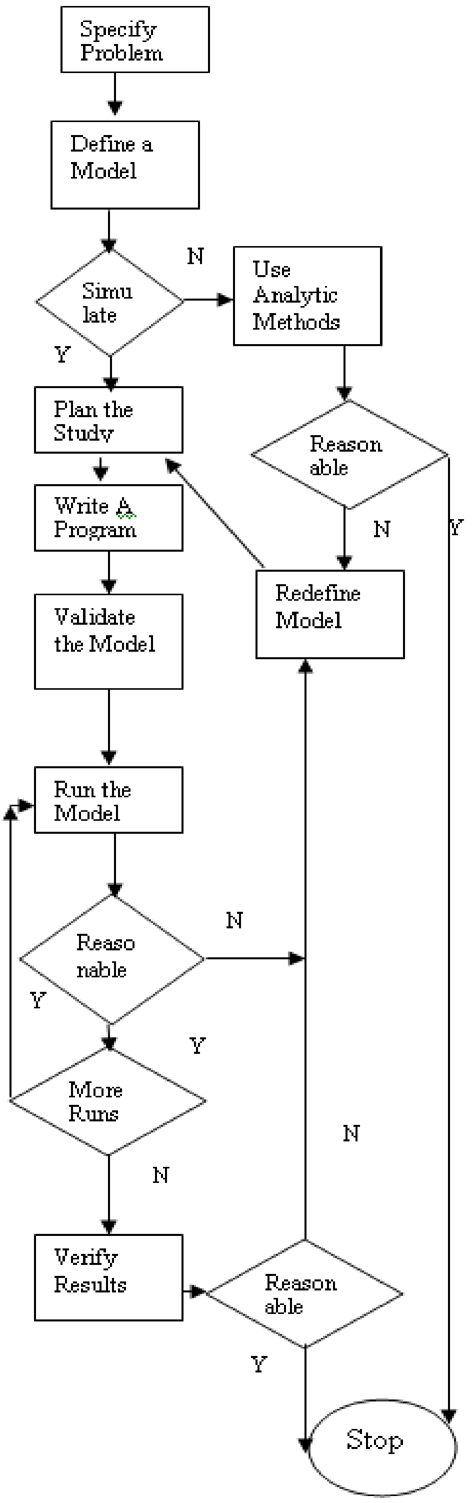
c) *Software Reliability Simulation Model*

A software reliability simulation model focuses on some particular software reliability estimation and/or prediction process vis-à-vis a software system. It can represent reliability of a software system as currently implemented (as-is), or as desired for future (to-be). Since all models are abstractions, a model represents only some of the many aspects of a software system that potentially could be modeled. This includes the aspects believed by the model developer to be especially relevant to the issues and questions the model is used to address.

IV. Simulation Process

Data has to be considered as real system or simulated data. The fact that good data sets are exactly scarce, one purpose of simulation is to supply carefully controlled, homogeneous data or software artifacts with known characteristics for use in evaluating the various assumptions upon which existing reliability models have been built. Since actual software artifacts (e.g. faults in computer programs) and processes (e.g. failure/fault removal) often violate the assumptions of analytic software reliability models, simulation can help a better understanding of such assumptions and may even lead to a better explanation of why some analytic models work well in spite of such violations[25][26][1]. Some of the steps involved in the process of simulation study [7], [1] are illustrated by the flowchart of Figure below (process of simulating).

©2011 Global Journals Inc. (US)



First up, it is required to describe the problem to be solved in a concise manner. Based on this problem definition, a model is defined. At this point of time, it can be checked out whether the model can be kept in a form that allows analytical techniques to be used. When it is decided to simulate, the experimental nature of the simulation technique makes it essential to plan the study by deciding upon the major parameters to be varied, the number of cases to be conducted and the order in which runs are to be made. Specification of experimentation has two components: experimental frame(s) and simulation run(s). An experimental frame defines a limited set of circumstances under which the system (or its model) is to be observed or subjected to experimentation. It requires specification of the observational variables, input schedules, initialization, and termination conditions and collection, compression, and display of data [20]. A simulation run is the observation of the behavior of a particular model under an experimental frame. Given that the simulation is to be on the digital computer, a program must be written. A program has both representation and execution aspects. Once the model is decided, we need to verify the model and then execute a series of runs according to the study plan. As results are obtained, it is likely that there will be many changes in the model and the study plan. The early runs may make parameter significance clear and so lead to the reassessment of the model. Verification of results is important after each run. Sometimes it is useful to repeat runs so that parts of model have different random numbers on each run. Moreover, one has to consider different groups of methodology or technique-oriented issues of modeling, experimentation, simulation, and programming. The degree of success of a simulation study is assessed in terms of the objective of the study, the structure and data of the real system, the parametric model, the model parameter set, the specification of experimentation, and the accepted norms of the modeling methodology, experimentation technique, simulation methodology, and software engineering.

V. Reliability Simulation Methodology

a) *Assumptions*

Assumptions and observed data are very important for software reliability study [3]. For the simulation (rate-based) we have the following assumptions. It may be noted that these assumptions can be seen as the most common assumptions for software reliability models [4], [22].

1. The software under testing remains essentially unchanged throughout testing, except for the removal of faults as they are found.
2. Removing a fault does not affect the chance that a different fault will be found.
3. "Time" is measured in such a way that testing effort is constant.
4. All faults are of equal importance.
5. At the time of testing, there is some finite total number of faults, which may be fixed or random; if random, their distribution may be known or of known form with unknown parameters.
6. Between failures occurrence, the failure rate follows a known functional form.

b) *Approaches To Reliability Simulation*

There are a number of modeling approaches used to investigate different aspects of the software reliability modeling process. The appropriate approach suited to the particular simulation model is best determined in terms of its purpose, questions, scope, result variables desired, etc. A variety of simulation approaches have been applied to software systems, which include: General discrete event simulation, System dynamics (or continuous simulation) and so on. However, the following are the two main approaches to reliability simulation.

i) *Rate-Based Reliability Simulation*

It is a rate-controlled event process simulation method. Here, a stochastic phenomenon is represented by a time sequence x(t), the behavior of which depends only on a rate function, R (t); R (t)*dt represents the conditional probability that a specified event occurs in infinitesimal interval (t, t+dt). Various mathematical reliability models work on Failure Rate Functions. The output of a rate-based reliability simulation approach is a time-line behavioral imitations of the activities and events involved in reliability. Reliability measures of interest in the software system are modeled parametrically over time. This approach is ba--sed on rate- based architecture, wherein phenomena occur naturally over time, controlled by their freque--ncies of occurrence. Software metrics such as number of faults so far exposed or yet remaining, failure criticality, test intensity, and software execution time govern the architecture. Rate based event simul--ation is an example of a form of modeling called system dynamics, whose distinctive feature is that the observables are discrete events randomly occurring in time. Since many software reliability growth models are based on rate (in terms of software failure/fault), the underlying processes/assumptions assumed by these models are fundamentally the same as the rate-based reliability simulation (see ASSUMPTIONS). In general, simulations enable investigations of questions too difficult to be answered analytically, and are therefore more flexible and more powerful.

ii) *Artifact-Based Reliability Simulation*

In this approach, many aspects of program construction and testing are used to investigate the effect of static features on dynamic behavior; Here, the inputs may include code structure characteristics, coding errors, test input data, test conduct, failure characteristics, debugging effectiveness, and computing environment. The output of this simulation approach is artifacts in an actual software environment according to factors and influences believed to typify these entities within a given context. The artifacts and environment are allowed to interact naturally, whereupon the flow of occurrences of activities and events is observed. This artifact-based simulation allows experiments to be set up to examine the nature of the relationships between software failures and other software metrics, such as program structure, programming error characteristics, and test strategies. It is suggested that the extent to which reliability depends merely on these factors can be measured by generating random programs having the given characteristics, and then observing their failure statistics.

A software system consists of static and dynamic structures with static structure existing in terms of component-interactions. It is evident by the inspection of the design and code of the software system and comprehendible without the need for its execution or simulation. However, it is the dynamic structure of the software system which is very important for reliability analysis. The dynamic/runtime information may include the frequency of occurrence of the interactions, the time spent in the interactions, etc. Dynamic structure is obtained by the execution or simulation of the software system. It depends on usage characteristics of the application, which is given by its operational profile [18].

VI. Implementing Reliability Simulation

With an intention to simulate the reliability measures of a software system, software system can be considered on a holistic basis in an approach called as black-box simulation or the software system can be considered as a bunch of some individual components/component combinations in another approach called as white-box simulation. Awareness of what to simulate is very important and can be helped by knowing (1) model scope, (2) result variables, (3) abstraction (represented by model), and (4) input parameters.

a) *Black-Box Reliability Simulation*

In black-box simulation approach to reliability, we treat software as a whole where only the application-level interfaces (input/out) hold significance, meaning that only the interactions with the outside world are modeled, while the internal structure and component combinations are not modeled. This is relatively a simple simulation approach. In black-box approach, only the failure data from the software systems under measurement are included in the modeling process, while the system structures are ignored. The input to the black-box simulation is a failure behavior file including the parameters of failure. The parameters can be obtained by using CASRE (Computer Aided Software Reliability Estimation) which is a tool for software reliability measurement [19]. The output of black-box simulation is the number of cumulative failures and the failure intensity of the software.

b) *White-Box Reliability Simulation*

In the black-box simulation, the software system is treated as a whole. The internal structure and features of software (e.g. the components interactions and correlations) are not concerned. There are some shortcomings in this approach for software reliability measurements analysis. With the increasing popularity of component-based software systems design, white-box approaches to software reliability seem more fitting for simulation. As a general practice, modeling is based on availability of the whole system data, without taking into account the unit testing data which is usually available earlier for each component. Also, the simulation process can be represented by one single model; however, it may be more appropriate that different components be applied different models. For such modeling considerations, white-box simulations are the solution. Generally speaking, the white-box approach to software reliability extends the black-box approach by including structural parameters into the reliability engineering process.

VII. What Does Simulation Offer

The immediate product of simulation study is a model that is primarily visual (i.e., graphical, diagrammatic, or iconic) or textual in form. Visual models have become the de facto standard for software systems simulations for their understandability and ease of development. Understanding is further helped when the model encompasses the ability to animate; the model during simulation can be used to show the flows of objects (e.g., code units, designs, problem reports) through the process, the activities currently being performed, and so forth. Moreover, a visual model is always supplemented by textual information regarding interrelationships among components, random variables distributions, etc. Thus, a model is genuinely desirable information bank.

The nature of simulation can be deterministic, stochastic, or hybrid. In the deterministic case, input parameters are spelled out as single values or point estimates (deterministic). Stochastic modeling encompasses the inherent uncertainty in many parameters and relationships. Stochastic variables are random numbers drawn from a specified probability distribution. Hybrid modeling employs both deterministic and stochastic parameters. In a purely deterministic model, only one simulation run is needed for a given set of parameters. However, with stochastic or hybrid modeling, the result variables or observables differ from one run to another because the random numbers actually drawn differ from run to run. Thus, the result variables are best analyzed statistically e.g., in terms of mean, standard deviation, distribution form etc. across a batch of simulation runs; this is termed Monte Carlo simulation.

Finally, sensitivity analysis, a very useful capability of simulation models, is used to understand and analyze the effects and/or the significance of effects caused by varying a selected model parameter in a controlled sense on some key observable. This allows the model developer to determine the likely range of results due to uncertainties in key parameters. It also allows the model developer to identify which parameters have the most significant effects on results, suggesting that those be measured and/or controlled more carefully.

VIII. Conclusion

It is crystal clear that simulation holds a lot of promise for modeling of software systems. Simulation techniques, applicable for the assessment of fully functional systems, can evaluate the reliability and performance, as early as the architecture phase in the life-cycle of the software. Thus, help in the selection of reusable components, identification of components that should be developed in-house, and allocation of reliabilities to the individual components so that the overall reliability objective is met. It can also help in the identification of reliability and performance bottlenecks, so that remedial actions can be taken before it is too late/ too expensive. Moreover, simulation lets us fore-see the working and behavior of a revised or new process, prior to its implementation, thereby averting expensive and risky process improvements through operational experience.

However, the effectiveness of simulation is guaranteed only if both the model, and the data driving the model, accurately reflect the real world. This emphasizes collection of metric data in a consistent sense from a systems perspective - it is not simply a collection of "nice to have" data. Usually, analyst does not have clear guidelines on what is essential metric data.

As a cautionary note, simulation is not a panacea. In fact, the predictive power of simulation is governed by model validation efforts. Simulation is a simplification of the real world, and is thus inherently an approximation. As indicated in [23] it is not possible that a model is absolutely correct. Therefore, model (verification and validation) is concerned with creating enough confidence in a model for its results to be accepted. This is done by trying to prove that the model is incorrect. The more tests that are performed in which it cannot be proved that the model in incorrect, the more confidence in the model is increased.

References Références Referencias

1. Quyoum, A., Dar, M., and Quadri, S.M.K., "Improving Software Reliability using Software Engineering Approach-A Review", International Journal of Computer Applications, November 2010.

2. Cheung, R. C., “A User-Oriented Software Reliability Model”. IEEE Transactions on Software Engineering, SE-6(2):118–125, March 1980.
3. Defamie, Patrick Jacobs, and Jacques Thollem. "Software Reliability: Assumptions, Realities and Data," Software Maintenance, 1999. (ICSM '99). Proceedings of IEEE International Conference on Software Maintenance, 1999, Page(s): 337 - 345
4. Dalal, S., Lyu, M.R., and Mallow, C., "Software Reliability," Chapter in Encyclopedia on Biostatistics, P. Armitage and T. Colton (eds.), vol. 5, Wiley 1998, pp. 4550-4555.
5. Eckhardt, D. E. Jr. and Lee., L. D., “A Theoretical Basis for the Analysis of Multiversion Software Subject to Coincident Errors”. IEEE Transactions on Software Engineering, SE-11(12):1511–1517, December 1985.
6. Farr, W., Handbook of Software Reliability Engineering, Lyu, M. R., Editor, chapter Software Reliability Modeling Survey, pages 71–117. McGraw-Hill, New York, NY, 1996.
7. Gordon,G., System Simulation.Prentice Hall of India,2010
8. Gokhale, S., Philip, T., and Marinos, P. N. , “A Non-Homogeneous Markov Software Reliability Model with Imperfect Repair”. In Proceedings of International Performance and Dependability Symposium, Urbana- Champaign, IL, September 1996.
9. Gokhale, S. , and Trivedi, K. S. “Structure-Based Software Reliability Prediction”. In Proceedings of 5th International Conference on Advanced Computing, Chennai, India, December 1997.
10. Gokhale, S. ,Lyu,M.R, and Trivedi, K. S. “Reliability Simulation of Component-based Software systems”. In Proceedings of 5th International Conference on Advanced Computing, Chennai, India, December 1997.
11. Hamlet, D., “Are We Testing for True Reliability?”. IEEE Software, 13(4):21–27, July 1992.
12. Horgan, J. R., and Mathur, A. P. , Handbook of Software Reliability Engineering, M. R. Lyu, Editor, chapter Software Testing and Reliability, pages 531–566. McGraw-Hill, New York, NY, 1996.
13. Laprie, J. C. , and Kanoun, K., Handbook of Software Reliability Engineering, Lyu, M. R., Editor, chapter Software Reliability and System Reliability, pages 27–69. McGraw-Hill, New York, NY, 1996.
14. Littlewood. B., “A Semi Markov Model for Software Reliability with Failure Costs”. In Proceedings of Symposium on Computer Software Engineering, pages 281–300, Polytechnic Institute of New York, April 1976.
15. Laprie, J. C., Kanoun, K., Beounes, C.,and Kaaniche, M., “The KAT (Knowledge-Action-Transformation) Approach to the Modeling and Evaluation of Reliability and Availability Growth”. IEEE Transactions. on Software Engineering, SE-17(4):370–382, 1991.
16. Littlewood, B., and Miller, D. R., “Conceptual Modeling of Coincident Failures in Multiversion Software”. IEEE Transactions on Software Engineering, 15(12), December 1989.
17. Musa, J.D., Iannino, A., and Okumoto, K., Software Reliability—Measurement, Prediction, Application. New York: McGraw Hill, 1987.
18. Musa, J.D. “Operational Profiles in Software-Reliability Engineering,” IEEE Software, vol. 10, no. 2, pp. 14-32, Mar. 1993.
19. Musa, J.D., and Okumoto, K., "A Logarithmic Poisson Execution Time Model for Software Reliability Measurement," Proceedings seventh International Conference on Software Engineering, Orlando, Florida, 1984, pp. 230-238.
20. Oren, T.I. and Zeigler, B.P. Concepts for advanced simulation methodologies. Simulation, 32, 3 (March 1979) pp. 69-82.
21. Parnas, D. L., “The Influence of Software Structure on Reliability”. In Proceedings of 1975 International Conference on Reliable Software, pages 358–362, Los Angeles, CA, April 1975.
22. Quadri, S.M.K., and Ahmed, N., “Software Reliability Growth Modeling with New Modified Weibull Testing-effort and Optimal Release Policy”, International Journal of Computer Applications, September 2011.
23. Robertson, S., "Simulation Model Verification and Validation: Increase the Users' Confidence", Proceedings of the 1997 Winter Simulation Conference, pp53-59.
24. Shooman. M. L., “Structural Models for Software Reliability Prediction”. In Proceedings of 2nd International Conference on Software Engineering, pages 268–280, San Fransisco, CA, October 1976.
25. Tausworthe. R.C., and Lyu. M.R, "A generalized technique for simulating software relibility,"IEEE
26. Software,”Vol.13, No.2, pp.77-88, March 1996. R.C.Tausworthe and M.R, Lyu. "Software Relibility Simulation,".
27. Tomek, L. A. et al, Muppala, J. K., and Trivedi. K. S., “Modeling Correlation in Software Recovery Blocks”. IEEE Transactions on Software Engineering, 19(11):1071–1086, November 1993.

©2011 Global Journals Inc. (US)

This page is intentionally left blank