iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** XII **Month of publication:** December 2017

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor :6.887
Volume 5 Issue XII December 2017- Available at www.ijraset.com

A Survey of Software Reliability Models

Dr. Shelbi Joseph[1], Akhil P V[2]
[1,2]*Information Technology, School of Engineering, Cochin University of Science and Technology*

Abstract: ***The reliability of computation systems involves failure-free operation of the software as well as hardware components. A lot of software models are available, and Software Reliability Engineering (SRE) is a skill to be more competitive in the environment of globalization and outsourcing. Customers want more reliable software that is faster and cheaper. SRE is a practice that is standard, proven, and widely applicable. Software development is very competitive and there are a lot of developers in a given domain. It is not sufficient that software works, but it is important that it meets the customer-defined criteria. Surveys reveal that the most important quality characteristics are reliability, availability, rapid delivery and low cost. Reliability suffers when it competes for attention against schedule and cost. In fact, this is the principal reason for well known existence of reliability problems in many software products. This establishes the significance of reliability estimation models. This paper reviews the various software reliability estimation models that were proposed in last few years. The review can be helpful for those who need a reference to start with in developing reliability models.***
Keywords: Software reliability, Computation model, Reliability estimation, Open Source Software, Hardware Reliability

I. INTRODUCTION

Reliability modelling is the process of predicting or understanding the reliability of a component or system prior to its implementation. Two types of analyses that are often used to model a complete system availability behaviour are Fault Tree Analysis and Reliability Block diagrams. The Reliability growth models are categorized as hardware models and software models. Hardware Reliability Growth Models (HRGM) is generally categorized as probabilistic models and statistical models. In probabilistic reliability growth models – because of no unknown parameters associated with these models, the data obtained during the program cannot be incorporated. Statistical reliability growth models – unknown parameters are associated with these models. In addition, these parameters are estimated throughout the development of the product in question.

In this paper, we will be discussing in detail the software reliability growth models, hardware reliability growth models and open source software reliability models. And finally a frame work is presented that enables the early prediction of software reliability. The paper will be a good reference material for various reliability estimation models spanned across hardware, software and open source softwares.

A. Reliability Growth Models

A Reliability growth model provides a systematic way of assessing and predicting system reliability based on certain assumptions about the fault in the system in a usage environment. It involves comparing measured reliability at a number of points of time, with known functions that show possible changes in reliability. A reliability growth model is a model of how the system reliability changes over time during the testing process. As system failures are discovered, the underlying faults causing these failures are repaired so that the reliability of the system should improve during system testing and debugging. To predict reliability, the conceptual reliability growth model must then be translated into a mathematical model. Reliability growth models can therefore be used to support project planning.

A variety of models are available for the estimation of reliability in the case of software as well as hardware. The role that software plays as a support to modern societal activities cannot be underestimated. However, the ability to predict software reliability is still not well understood and it needs further study. Although a number of software reliability models have been developed till date, none has been universally accepted in the field [1] [2].

Taxonomy of reliability models is as shown in the Fig 1. The figure shows present hardware, closed source software and open source software reliability models. The hardware reliability models include Weibull model, Constant hazard model and linearly increasing model. The closed source software models are generally classified as failure rate models and Non Homogeneous Poisson Process (NHPP) models. The failure rate models are again divided as general and Bayesian models. There are assessment and predictive models in the general category. In the case of OSS there are certain studies which conclude that the Weibull distribution can be used as a model.

The taxonomy presentation gives an abstract view of the various models available for estimation of reliability of both hardware and software. It also summarises the significant works that were proposed in the previous decades. The works are also classified on the basis of its predictive nature and assessment capability.

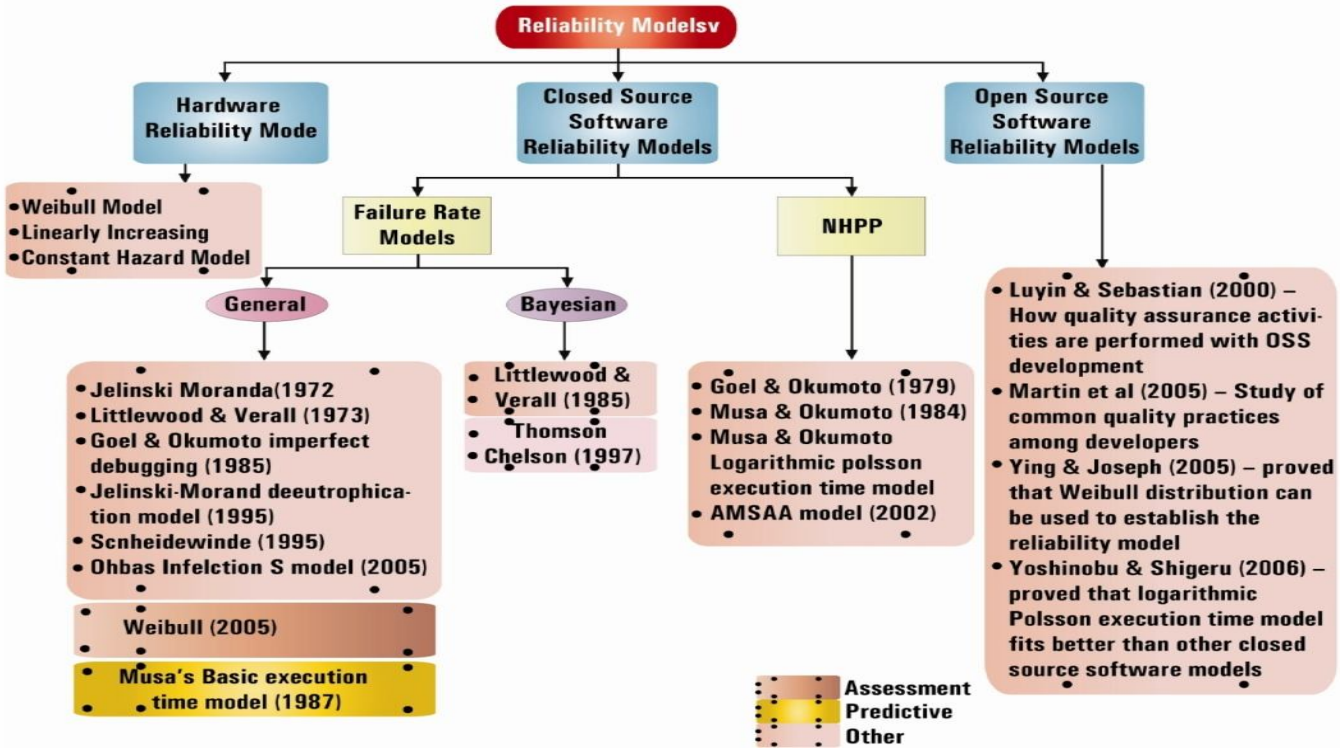


Figure 1: Taxonomy of Software Reliability Models

II. RELIABILITY MODELS

A. Software Reliability Growth Models

A systematic frame work designed to predict software reliability from software engineering measures was summarized as follows [3], [2]. Research activities in software reliability engineering have been conducted over the past two decades and many Software Reliability Growth Models (SRGMs) have been proposed for the estimation of software reliability and number of faults remaining in the software [4], [5], [6], [7], [8], [9]. Most of the SRGMs assume that each time a failure occurs, the error which caused it, is immediately removed and no new errors are introduced [10].

Most models make some assumptions about the software failure process so that the model becomes mathematically tractable [11],Goel [12] has given the typical assumptions made in Software Reliability Model with its limitations. Reliability models have been proposed by Goel and Okumoto [13], Jelinski and Moranda [14] ,Little wood and Verrall [15] Musa and Okumoto [16], and Shooman [17]. To employ a model for reliability prediction, value of some of the parameters need to be specified. These are typically determined by analyzing the past failure data of the software. The J-M model proposed by Jelinski and Moranda [18] is one of the simplest and earliest of the software reliability models. The J-M model assumes that times between failures are independent random variables following exponential distributions, there are finite number of faults at the beginning of the test phase, and that the failure rate is uniform between successive failures and is proportional to the current error content(number of faults remaining) of the program being tested. This model is very simple to use. It is also fairly accurate for some data sets, but sometimes leads to inaccurate predictions. (Pankaj and RajibGhosh [19]).

The basic execution model proposed by Musa et. al.[20] make assumptions similar to the above model except that the process modelled isthe number of failures in specified execution time intervals. There are a finite number of faults in the beginning of the test phase, and the times between failures are exponential, the failure rate being uniform between successive failures. He also provides a systematic approach for converting the model so that it can be applicable for the calendar time as well. The Little wood and Verrall model [21] assumes exponential distribution for the random variable representing the failure interval time. But the failure intensity is regarded as a stochastically decreasing function with gamma distribution, implying that the fault fixing process is

not considered as perfect, and that faults are of different sizes. A user controlled function determines the nature of the reliability growth. This model requires complex statistical inference for determining the parameters.

The Goel and Okumoto (G-O) model [22] considers the software failure process as a Non Homogeneous Poisson Process (NHPP) with a mean function μ(t). This model treats initial error contents as a random variable. The M-O model proposed by Musa and Okumoto [23] views failure process as a NHPP like G-O model. But Unlike G-O model it assumes reduction in failure rates are greater for the earlier fixes. MO model assumes failure rate to be an exponential function of the expected number of failures. Input to the model is in the form t1,t2,……,where each tj represents the execution time. Execution time is related to calendar time through some suitable assumption and further computation.

As we can see, the basic input to all these models is the times of past failures, or times between consecutive failures. These data are used to determine the value of parameters, and then to predict the reliability of the given software. Most of the models use calendar time, and where execution time is used, suitable methods are used to convert it to calendar time.

Goel and Okumoto proposed an imperfect debugging model called Goel and Okumoto Imperfect Debugging Model (Amrit and Goel[24]), which assumes that faults are removed with certainty when detected, is not always the case. In this model, the number of faults in the system at time t, X(t), is treated as a Markov process whose transition probabilities are governed by the probability of imperfect debugging. Times between the transitions of X(t) are taken to be exponentially distributed with rates dependent on the current fault content of the system.

B. Hardware Reliability Growth Models

Understanding the dynamic behaviour of system reliability becomes an important issue in either scheduling the maintenance activities or dealing with the improvement in the revised system design. In doing so, the failure or hazard rate function should be addressed. Bathtub curve is usually adopted to represent the general trend of hazard rate function. Many studies were concentrated on depicting the geometric shape of the bathtub curve. The early contributors in this area include , Bain [25], Smith & Bain [26], Gaver [27], Hijroth [28], Dhillon[29], Lawless [30], Jaisingh et. al. [31], Haupt&Schabe [32], Schabe [33], Xie and Lai [34], Edelstein [35]. [36]proposed a general form of bathtub shape hazard function in terms of reliability. The relation between hazard rate and reliability of a system follows the definition [36].

$$Z(t) = -\frac{1}{R(t)}\frac{dR(t)}{dt}$$

Usually the reliability decreases monotonically with time and thus there is a one to one correspondence between reliability and time. That is, the hazard rate function can also be expressed as

$$Z(t) = -\frac{1}{R(t)}\frac{1}{dt/dR(t)} = Z(R)$$

Thus, instead of the usual procedure of estimating Z(t) the relationship of Z(R) based on the available data was defined. The change of expression Z (t) to Z(R) has certain advantages. First, the equation of dynamic reliability takes an autonomous form; particularly it belongs to a general type of logistic equation encountered very often in ecological science (Edelstein [37]). Therefore good experience can be guided from these studies. Secondly, the hazard rate is investigated in finite domain (1, 0) as compared with that in infinite domain of time sequence. Wang et. al. [38] developed reliability models that can be applied for the development of a new mechanical product with modified function requirements. Wang et. al. [39] also developed reliability models for material fracture due to crack growth. The data obtained from failure tests can be analysed to obtain reliability, failure density, hazard rate and other necessary information (Srinath [40]). Obviously, the behavioural characteristics exhibited by one class of components differ from those exhibited by another class of components. In order to compare different behavioural characteristics and also to draw general conclusions from behavioural patterns of similar components, a mathematical model representing the failure characteristics of the components becomes necessary. The procedure involves assuming a function for hazard rate, and thereby obtaining reliability and failure density by using this failure rate function. The assumed function for the hazard rate will be the hazard model.

C. Reliability Models for Open Source Software

The OSS development mainly depends on the practice of welcoming every enthusiastic individual who would like to contribute to the project. On top of this, the freedom of using, modifying and distributing OSS leads to more robust software and more diverse business models (Wu and Lin [41]). Software reliability models are useful to assess the reliability for quality management and testing progress control of software development. Although open source practices have been remarkably successful in recent years, the open source development model faces a number of product quality challenges. Rare open source projects have been archived

successfully as a high level quality end product. However, these mature and successful projects face quality problems too. Even though lots of models and tools have been suggested for reliability checking, very few models are applied and tested in this case.

One of the recent studies by CoverityInc [http://www.coverity.com] on measuring reliability of open source software claims that the LAMP stack – Linux, Apache, MySQL, and Perl/PHP/Python – showed significantly better software quality above the baseline with an average of 0.290 defects per thousand lines of code, compared to an average of 0.434 for the 32 open source software projects analysed. One of their goals of research on software quality was to define a baseline so that people can measure software reliability in both open source and proprietary software projects.

that OSS development is very different from the traditional software development used in most of the software industry. Moreover, the quality assurance activities are also performed in a different fashion. Martin et.al. [43] have done exploratory interviews with free and open source developers to study the common quality practices among the developers to implement a quality process improvement strategy. They found that even though development of OSS projects share common practices the quality of the resulting products needs further empirical evaluation. This implies that we have to look into reliability models for open source software development.

An empirical study towards open source software reliability model was conducted by Ying and Joseph [44]. They have collected data from eight active open source projects from “SourceForge.net” and reliability analysis was done based on the bug arrival rate. They claim that general Weibull distribution is a possible way to establish the reliability model. Further, in contrast with closed source projects, it is unlikely to find a Rayleigh curve, to model all open source projects. In a recent study on Xface desktop environment, an open source distributed project, Yoshinobu and Shigeru [45], attempted an evaluation under Mozilla public license by applying various reliability growth models. Conventional models like exponential growth model, delayed S-shaped model, inflection S-shaped model and logarithmic Poisson execution time model were considered and goodness-of-fit comparison were done. Various software reliability assessment measures were derived from the non homogeneous Poisson process (NHPP) models. It has been concluded that the logarithmic Poisson execution time model fits better than the other software reliability growth models for the actual data set.

D. Computational System Reliability

Computational system reliability is concerned with hardware reliability, software reliability, reliability of interaction between hardware and software and reliability of interaction between the system and the operator. In general, a system may be required to perform various functions, each of which may have a different reliability. In addition, at different times, the system may have a different probability of successfully performing the required function under stated conditions. The analysis ofthe reliability of a system must be based on precisely defined concepts.

Software intensive systems are increasingly used to support critical business and industrial processes, such as in business information systems, e-business applications, or industrial control systems. Reliability engineering gains its importance in the development process. Reliability is compromised by faults in the system and its execution environment, which can lead to different kinds of failures during service execution: Software failures occur due to faults in the implementation of software components, hardware failures result from unreliable hardware resources, and network failures are caused by message loss or problems during inter component communication (Franz and Heiko [46]).

The analysis of the reliability of a system must be based on precisely defined concepts. Since it is readily accepted that a population of supposedly identical systems, operating under similar conditions, fall at different points in time, then a failure phenomenon can only be described in probabilistic terms. Thus, the fundamental definitions of reliability must depend on concepts from probability theory (Pham [47]).

System-level reliability and availability requirements set forth by U.S. Government agencies procuring large software intensive systems encompass both hardware and software. However, specifications, statement of work requirements, and compliance documents (standards) usually implicitly or explicitly focus on hardware and are largely silent about software reliability, maintainability, availability and dependability. Consequently, contractor system reliability analyses and design reviews usually ignore quantitative software reliability, maintainability, availability, and dependability requirements. During system testing and evaluation, data on software operating times, failure rates, and recovery times are not collected. Finally, logistics and support specialists devote significant attention to sparing and maintenance concept development, but often do not adequately consider the software-related sustainment issues of large computer systems. These problems can be solved, by an appropriate definition of requirements for software-intensive system reliability in specifications, and in the definition of programmatic requirements in contractual documentation (Myron et. al.[48]).

In general, a system may be required to perform various functions, each of which may have a different reliability. In addition, at different times, the system may have a different probability of successfully performing the required function under stated conditions. The term failure means that the system is not capable of performing a function when required. The term capable used here is to define if the system is capable of performing the required function. However, the term capable is unclear and only various degrees of capability can be defined (Musa [49], Pham [47]).

III.FRAMEWORK FOR RELIABILITY PREDICTION

A. A Framework to Enable the Early Prediction of Software Reliability

The objective is to develop a framework to enable the early prediction of software reliability incorporating reliability measurement in each stage of the software development. Leslie et.al.[50] state that the ability to predict the reliability of a software system early in its development can help to improve the systems quality in a cost effective manner. Therefore, the proposed framework measures and minimizes the complexity of software design at the early stage of software development lifecycle, leading to a reliable end product. To calculate the reliability of software product, the reliabilities at different stages of product development like requirements analysis, design, development, testing and implementation etc. will have to be evaluated. This facilitates the improving of the overall product reliability. It is observed that modifications and error identifications during operation and implementation can lead to reengineering of large parts of the system, which has been shown to be costly. Hence to ensure the quality of the developed system, it is important to ensure quality at different stages of development. A few approaches which do consider component-level reliability (Goseva et. al. [51], Reussner et, al. [52]) , assume that the reliabilities of a given component's elements, such as its services, are known. Reliability prediction is useful in a number of ways. A prediction methodology provides a uniform, reproducible basis for evaluating potential reliability during the early stages of a project. Predictions assist in evaluating the feasibility of proposed reliability requirements and provide a rational basis for design and allocation decisions.

B. Reliability Prediction

Reliability predictions are conducted during the requirement and definition phase, the design and development phase, the operation and maintenance phase in order to evaluate, determine and improve the dependability measures of an item. Successful reliability prediction generally requires developing a reliability model of the system considering its structure. Several prediction methods include reliability block diagrams, fault tree analysis, state-space method etc.

A prediction scenario is shown in the Fig 2. The method involves collection of failure data from the field and it is compared with the information available in the database. The database is updated regularly to keep it current. The failure rate figures are employed tested and checked in some suitable reliability models to predict the reliability. This will help the project managers to predict and develop a reliable product.

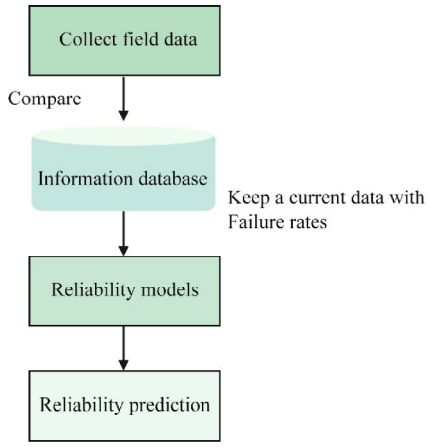


Figure 2: Prediction Scenario

C. The Framework

The main task of a system program office (SPO) when acquiring a new software system is to specify the requirements to the developer and to see that the requirements mentioned are met as the system development process evolves to the final product. It is also necessary to assure that the qualities of the software such as reliability, maintainability, usability, testability, and portability are attained.

An error in the software product can occur when there is a difference between the actual output of the software and the expected correct output. Fault is a condition that causes a system to fail to perform its required function. Failure occurs when the behaviour of the software is different from the specified behaviour (Amitabha and Khan [53]).

A reliability prediction framework is shown in the Fig 3, which is to analyse the reliability at different stages of development. he process includes phases of software development, identification of errors, integration, development and finalization. The first step is to analyse the phases of development, which includes requirement analysis, design, coding, testing, implementation and maintenance. Next comes the identification of errors in different phases, where possible occurrences of errors are identified. The collected error data is used to calculate the failure density and thereby the reliability.

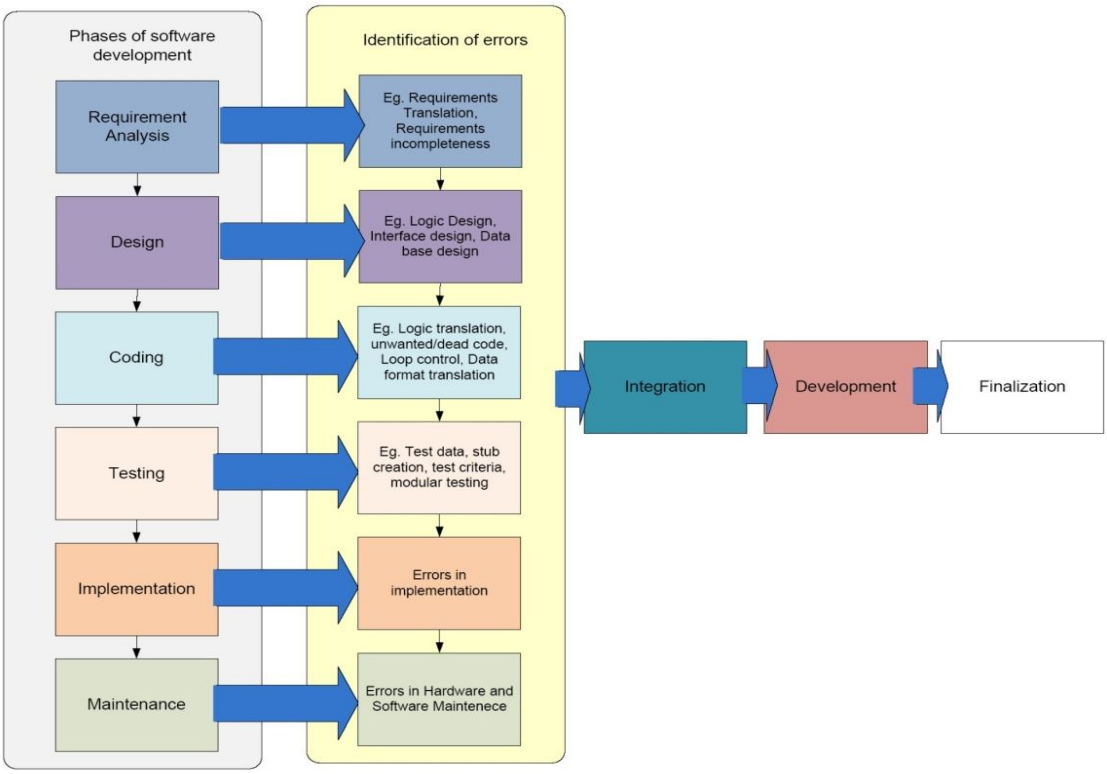


Figure 3: Reliability Prediction Framework

In the Identification phases, Software reliability attributes have been identified in different phases of development. Firstly, draw up a functional profile, then identify the needs of software reliability, then define the fault/failure type and the fault/failure severity, finally, understand the software development process and environment. Integration phase relation between reliability aspects of the above identified phase is determined. The next stage is to formulate a plan to integrate the software aspects to incorporate reliability criteria in the software development stage.

Reliability estimation model (REM) is developed in the development phase. In this phase, first of all, establish a data collection plan and collect data through templates. Secondly, draw up an operational profile and allocate software reliability goal. Predict software reliability through software prediction models and estimate software reliability through software estimation models. Elicit improvements and review improvements and establish a device for software reliability improvement (Voas [54]). Finally on the basis of the review, the whole approach is reviewed and revised if needed.

IV. CONCLUSIONS

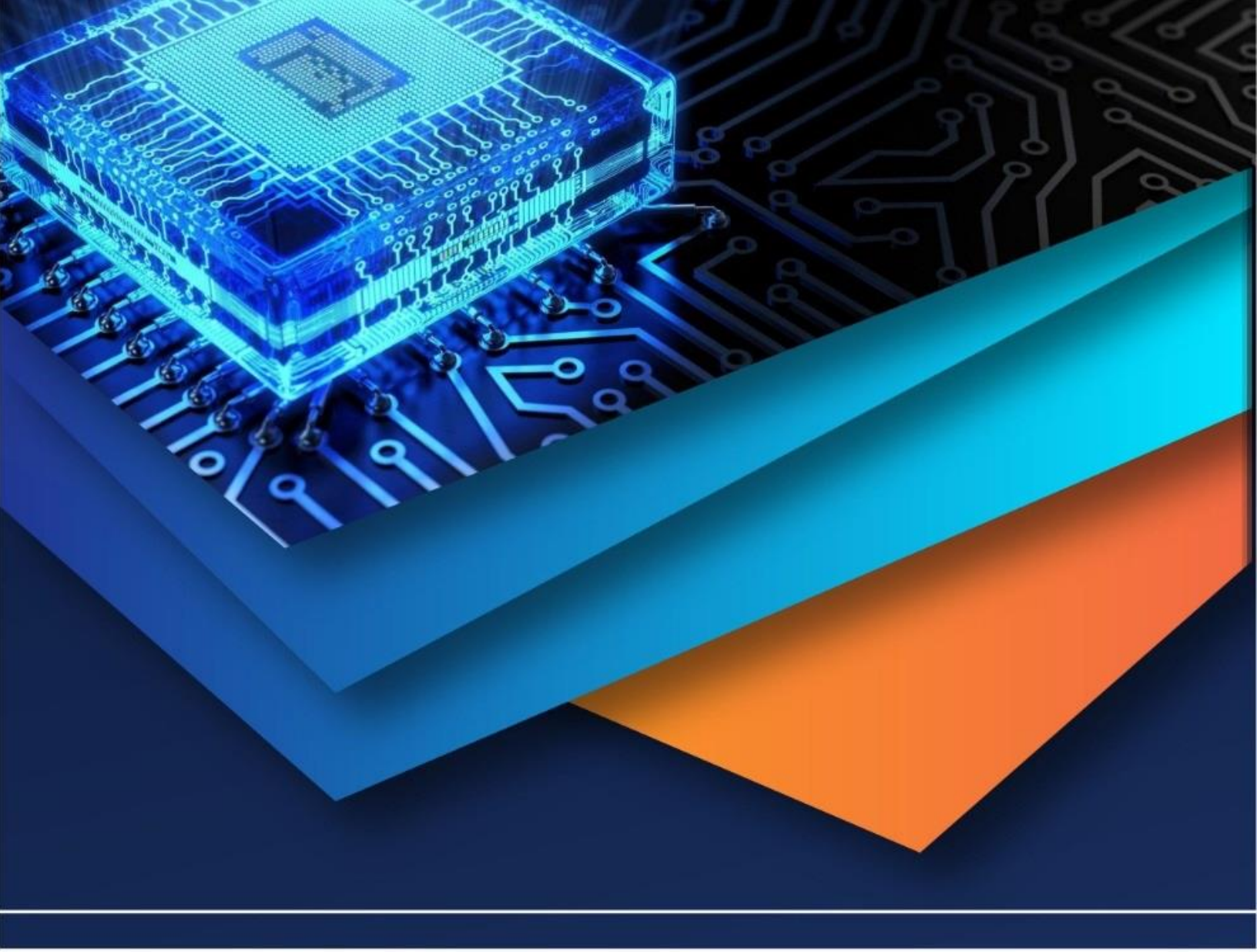
From the review of different models presented above it is evident that even though a lot of models are developed and available in the literature for evaluation of software reliability, all the models do not provide a direct quantification of the reliability, that is, all these models are not necessarily deterministic. Typical hardware reliability models make use of the available component field failure data for reliability estimation. However no attempts were made to incorporate hardware and software together in reliability estimation and the present work is emphasized on this. Also in the early stages of development, failure information is not available to quantitatively measure reliability of a software product. Software reliability cannot be calculated during the requirement analysis,

design, development, testing and maintenance phases, if adequate data on system failures is collected throughout the project. The same models for estimating reliability parameters, such as the expected number of failures in a certain period of time, failure intensity, the expected time of the next failure, etc., could be applied to software systems as well. A software reliability prediction framework is proposed, which enhances the reliability calculation at different stages of development and hence increases the end product reliability.

REFERENCES

[1] SmidtsC ,B.Li, and Z.Li, "Software Reliability Models, " in Encyclopaedia of Software Engineering, vol . 2, J.J. Marciniak, Ed.,2nd ed. New York: John Wiley & sons inc.,2002,pp. 1594-1610.
[2] Li M. and C.Smidts,' A Ranking of Software Engineering Measures based on Expert Opinion," IEEE Transactions on Software Engineering, vol. 29,pp, 811-24, 2003.
[3] Smidts C and M.Li, "Software Engineering Measures for Predicting Software Reliability in Safety Critical Digital Systems, "University of Maryland, Washington D.C.NUREG/GR-0019, November 2000.
[4] Goel A.L. and K.Okumoto, "Time-Dependent Error-Detection Rate Model for software and other performance measures," IEEE Trans. Reliability, vol. 28, pp. 206-211, 1979.
[5] Hossain S.A and R.C.Dhahiya, "Estimating the parameters of a Non-Homogeneous Poisson Process Model for software Reliability Model for Software Reliability," IEEE Trans. Reliability, vol.42, pp. 604-612, 1993.
[6] Leung Y.W., "Optimal Software Release Time with a given Cost Budget," J. Systems and Software, vol. 17. Pp. 23-242, 1992.
[7] Ohba M., "Software Reliability Analysis Models," IBM J. Research and Development, vol 28, pp. 428-443, 1984.
[8] Pham H, "Software Reliability Assessment: Imperfect Debugging and Multiple Failure Types in Software Development,"EG&GRAAM- 10737, Idaho Nat'1 Eng. Laboratory,1993.
[9] Yamada S. and S.Osaki, "Software Reliability Growth Modeling: Models and Applications," IEEE Trans. Software Eng., vol. 11, pp.1,431-1,437,1985.
[10] Hoang Pham, Xuemei Zhang, "A Software Cost Model with Warranty and Risk Costs,"IEEE Transactions on Computers, vol. 48, No.1, January 1999.M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in Proc. ECOC'00, 2000, paper 11.3.4, p. 109.
[11] PankajJalote, and RajibGhosh, "An Approach for Cost Effectiveness Analysis of Multiversion Software Using Software Reliability Models" http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.3640.,2011
[12] Goel .A.L, "Software reliability models: Assumptions,limitations, and applicability", IEEE Trans. Software Engineering,vol SE-11, num 12, 1985, pp 1411 - 1423.
[13] Goel A.L. and K.Okumoto, "Time-Dependent Error-Detection Rate Model for software and other performance measures," IEEE Trans. Reliability, vol. 28, pp. 206-211, 1979
[14] Z. Jelinski and P.B. Moranda,"Software reliability research", in Statistical Computer Performance Evaluation, W. Freiberger, Ed., New York: Academic Press, 1972, pp. 465-484
[15] Littlewood B. and J.L. Verrall, A Bayesian Reliability Growth Model for Computer Software, J. Royal Statist. Soc., C (Applied Statistics), Vol. 2, pp 332-346, 1973.
[16] J.D. Musa and K. Okumoto, "A logarithmic poisson execution time model for software reliability measurement", 7th Int'l Conference on Software Engineering (ICSE), 1984, pp. 230-238
[17] Shooman M.L, "Probablistic Methods For Software Reliability Prediction", Statistical Computer Performance Evaluation, Academic Press, New York, June 1972, pp 485-502.
[18] Moranda.P.L. and Jelinski. Z.,Final Report on Software Reliability Study, McDonnell Douglas Astronautics Company, MADC Report Number 63921,1972
[19] PankajJalote, and RajibGhosh, "An Approach for Cost Effectiveness Analysis of Multiversion Software Using Software Reliability Models" http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.3640.,2011
[20] Musa J.D. and A. Iannino and K. Okumoto, Software Reliability, Measurement, Prediction, Application, McGraw-Hill, 1987.
[21] Littlewood B. and J.L. Verrall, A Bayesian Reliability Growth Model for Computer Software, J. Royal Statist. Soc., C (Applied Statistics), Vol. 2, pp 332-346, 1973.
[22] Goel A.L. and K.Okumoto, "Time-Dependent Error-Detection Rate Model for software and other performance measures," IEEE Trans. Reliability, vol. 28, pp. 206-211, 1979.
[23] J.D. Musa and K. Okumoto, "A logarithmic poisson execution time model for software reliability measurement", 7th Int'l Conference on Software Engineering (ICSE), 1984, pp. 230-238.
[24] Amrit L and Goel. Software reliability Models: Assumptions, Limitations, and Applicability. Transactions on Software Engineering, Vol. 2, No. 12, pp 1411-1423, 1985 IEEE.
[25] Bain, I. J. (1974) Analysis of linear failure rate life testing distribution. Technometrics 16:551-60
[26] Smith, R. M. and Bain I. J. (1975) An exponential power life-testing distribution. Communications In Statistics 4(5): 469-81.
[27] Gaver, D. P. and M. Acar. (1979) Analytical hazard representation for use in reliability, mortality and simulation studies. Communications in Statistics-Simulation and Computation B8 (2):91-111.
[28] Hijroth, U. (1980) A reliability distribution with increasing, decreasing, constant and bath-tub shaped failure rates. Technometrics 22(1):99- 107.
[29] Dhillon, B. S. (1981) Life distributions. IEEE Transactions on Reliability R30:457-9.
[30] Lawless, J. F. (1982) Statistical models and methods for life time data. Wiley, New York.
[31] Jaisingh, L. R., W. J. Kolarik and D. K. Dey. (1987) A flexible bathtub hazard model for non-repairable systems with uncensored data. Microelectronics reliability 27(1): 87-103.
[32] Haupt, E. and H. Schabe. (1992) A new model for life time distribution with bathtub shaped failure rate. Microelectronics Reliability 32(5):633-9.

[33] Haupt, E. and H. Schabe. (1992) A new model for life time distribution with bathtub shaped failure rate. Microelectronics Reliability 32(5):633-9.
[34] Xie, M. and C. D. Lai. (1996) Reliability analysis using an additive Weibull model with bathtub shaped failure rate function. Reliability Engineering and System Safety 52: 87-93.
[35] Edelstein, K. L. (1998) Mathematical models in biology. Random House, New York.
[36] K.S. Wang, F.S. Hsu and P.P. Liu "Modeling the bathtub shape hazard rate function in terms of reliability", Reliability Engineering & System Safety Volume 75, Issue 3, March 2002, Pages 397–406.
[37] Edelstein, K. L. (1998) Mathematical models in biology. Random House, New York.
[38] Wang, K. S., E. H. Wan and W. C. Yang. (1993) A preliminary investigation of new mechanical product development based on reliability theory. Reliability Engineering and System Safety 40: 187-94.
[39] Wang, K. S., S. T. Chang and Y. C. Shen. (1996) Dynamic reliability models for fatigue crack growth problem. Engineering Fracture Mechanics 54(4): 543-56.
[40] Srinath LS. Reliability engineering. 3rd ed. New Delhi:Affiliated East West Press;1991.
[41] Ming-Wei Wu, Ying Dar – Lin, Open Source Software Development: An Overview. Computer, 34(6) 33 - 38, June 2001.
[42] Luyin Zhao and Sebastian Elbaum A Survey On Quality Related Activities in Open Source; Software Engineering Notes Vol 25 no 3: 54-57 May 2000.
[43] Martin Michlmayr, Francis Hunt, David Probert Quality Practices and Problems in Free Software Projects, Proceedings of the First International Conference on Open Source Systems Genova, pp 24-28, July 2005
[44] Ying Zhou and Joseph Davis Open source software reliability model: an empirical approach, Fifth workshop on Open Source Software Engineering (5-WOSSE) May 2005, USA.
[45] Yoshinobu Tamura and Shigeru Yamada, Comparison of Software Reliability Assessment Methods for Open Source Software and Reliability Assessment Tool, Journal of Computer Science vol 2 (6): pp 489-495, 2006.
[46] Franz Brosch, HeikoKoziolek "Architecture-Based Reliability Prediction with the Palladio Component Model" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 38, NO. 6, NOVEMBER/DECEMBER 2012.
[47] Pham H. "System Software Reliability" http://www.springer.com, , XIV,440 p.63 illus., Hardcover ISBN, 2007.
[48] Myron Hecht, Karen Owens, Joanne Tagami "Reliability-Related Requirements in Software-Intensive Systems" 2007 IEEE.
[49] Musa J.D "Meassurement and Management of Software Reliability" proceedings of the IEEE, VOL. 68, NO. 9, September 1980.
[50] Leslie Cheung, RoshanakRoshandel, NenadMedvidovic, LeanaGolubchik ",Early Prediction of Software Component Reliability" ICSE' 08, May 10 - 18, 2008, Leipzig, Germany.Copyright 2008
[51] Goseva-Popstojanova K. et al. Architectural Level Risk Analysis using UML, IEEE Transactions on Software Engineering,Vol.29, No.10, October 2003.
[52] Reussner R. et al. Reliability prediction for component-based software architectures, J. Systems and Software, 66(3), 2003.
[53] AmitabhaYadav and R.A. Khan "Reliability Estimation Framework- Complexity perspective" ICAITA, SAI, SEAS, CDKP, CMCA, CS & IT 08, pp. 97–104, 2012. © CS & IT-CSCP 2012.
[54] Voas,J., Payne,J.: Dependability certification of Software Components, Journal of Systems and Software 2000, 52(2-3) pp.165-172.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY