ISSN 2250-3153

Taxonomical study of Software Reliability Growth Models

Tariq Hussain Sheakh[1], VijayPal Singh[2]

[1]Lecturer in computer sciences at Govt. Degree College Poonch, J&K,INDIA
[2]Assistant Professor in Computer Science at JJT University Rajasthan

Abstract- Reliability has convert windfall of the Software without which we cannot even contemplate of it. Reliability is very imperative which amplify the software endurance and its longevity. Due to the augmentation of the complexity and the complex system the demand of the reliability becomes imperative. Reliability growth models preempt and derived the consistency of the software which becomes obligatory. Now –a-days nearly to hundred models for reliability have been developed. But research Scholar has difficult to choose it for reliability measure. In our research paper major emphasis is towards the taxonomical review of the reliability growth model and analyzing reliability pattern which is utterly based upon literature survey.

Index Terms- Software Reliability, Reliability Models, Reliability Modeling, Model's Taxonomy.

I. INTRODUCTION

Towards moving 21's century, software becomes a coercer for everything from elementary education to genetic engineering. Dependency & requirements on computer increases the difficulties & failures. Due to increase in addiction the size & complexity of system has grown. To avoid the failures & faults, reliability of software needs to be study during development of software so as to come up with reliable software. There are several projects executed by NASA, & DOD that deal with highly complex software. Due to change in the three lines of code in a single program in 1991 the telephone system was collapsed in California and eastern parts. Because of software failure, aircraft industry also faced lots of airliner crashes and abnormal flight conditions due to incompatible response to the pilots. This paper presents a review on the software reliability models. The study throws the light on various dimensions of reliability models. Categorization of models is based on identified dimensions.

A. Software reliability

Software reliability is a key attribute to software quality. Reliability is the property of referring 'how well software meets its requirements & also 'the probability of failure free operation for the specified period of time in a specified environment'. Software reliability defines as the failure free operation of computer program in a specified environment for a specified time Software Reliability is an important to attribute of software quality, together with functionality, usability, performance, serviceability, capability, install ability, maintainability, and documentation. Software Reliability is hard to achieve, because the complexity of software tends to be high. While any system with a high degree of complexity, including software, will be hard to reach a certain level of reliability, system developers tend to push complexity into the software layer, with the rapid growth of system size and ease of doing so by upgrading the software. While the complexity of software is inversely related to software reliability, it is directly related to other important factors in software quality, especially functionality, capability, etc. Emphasizing these features will tend to add more complexity to software.

B. Software reliability modeling

Among all software reliability models, SRGM is probably one of the most successful techniques in the literature, with more than 100 models existing in one form or another, through hundreds of publications. In practice, however, SRGMs encounter major challenges. First of all, software testers seldom follow the operational profile to test the software, so what is observed during software testing may not be directly extensible for operational use. Secondly, when the number of failures collected in a project is limited, it is hard to make statistically meaningful reliability predictions. Thirdly, some of the assumptions of SRGM are not realistic, e.g., the assumptions that the faults are independent of each other; that each fault has the same chance to be detected in one class; and that correction of a fault never introduces new faults. Although some historical SRGMs have been widely adopted to predict software reliability, researchers believe they can further improve the prediction accuracy of these models by adding other important factors which affect the final software quality. Among others, code coverage is a metric commonly engaged by software testers, as it indicates how completely a test set executes a software system under test, therefore influencing the resulting reliability measure. To incorporate the effect of code coverage on reliability in the traditional software reliability models, proposes a technique using both time and code coverage measurement for reliability prediction. It reduces the execution time by a parameterized factor when the test case neither increases code coverage nor causes a failure. These models, known as adjusted Non-Homogeneous Poisson Process (NHPP) models, have been shown empirically to achieve more accurate predictions than the original ones.

II. SOFTWARE RELIABILITY WORK

Generally, Software reliability is working on three phases as 1. Problem searching ie. To catch the faults or errors.2. Applying Techniques in order to solve or remove the problem. 3. Verifying and validating these techniques Reliability Modeling

1. Problem Searching

This is the first phase of software reliability plan. In this phase we are applying testing and comparing the performance of software with its previous standards. By using different type of testing such as black box and white box testing, we catch the problem occurring. If we catch the problem then we go to next phase as TMA for predicting, removing & solving the problem.

2. Applying Methods

After successful completion of problem searching phase we are applying the Methods. Methods is the collection of technique, approaches and methods. Here a lot of techniques, methods (model as SR model of Goel, Okumoto, Musa etc is used.) by using these models we can solve the upcoming problem in our software. We also apply few important approaches for solving these problems.

Approaches: - For selecting the SR model, to predict the software from failure & faults, two important approaches are (a) A model selection approach classified SR model on the basis of SDLC. Means we first check the status of upcoming problem i.e., in which phase of SDLC the problem is occurred. Then we apply related SR model for that phase to predict the software. (b) Systematic approaches of SR modeling are distinguished between SR model and parameters estimations procedure by using predicting performance. (ii) Techniques:-Generally in searching the problem, two testing techniques are applied. These techniques are (a) Black box testing (b) White box testing (iii) Methods: - By the help of methods we are finding approaches and techniques. It means they are inter-related to each other.

Fig-1: Reliability models classification

Models	Proposed by	Year	Type
JMM	Z. Jelinski Paul and B. Moranda	1972	Binomial
GM	Amrit Goel and Kazu Okumoto	1979	Poisson
ETM	John Musa	1975	
HEM	Ohba	1984	
WM	Weibul	1983	Binomial
SSM	S. Yamada, M. Ohba, and S.Osaka	1983	Gamma
DM	J.T Duane	1964	
GM	Paul B. Moranda	1979	Binomial
LPM	Musa-Okumoto	1984	Binomial
LVRG	A.Ghaly, P. Chan, & B. Littlewood	1986	Gmma

There are numbers of software available which can mimic the process involved in your research work and can produce the possible result.

A. Exponential Failure Time Model (EFTM) Exponential models comprise of all finite failure models. Poisson and Binomial are two categorization of EFTM the Binomial and Poisson types are based on per fault constant hazard rate. Hazards rate function is defined as the function of the remaining number of faults and the failure function is exponential. H (Z) = f (RNF) +f((exp(FF)) (1) Where, H (Z) = Hazard rate.RNF=Renaming number of faults. FF= Failure Function.

- J -M Model (JMM): The failure time is proportional to the remaining faults and taken as an exponential distribution. During testing phase the number of failures at first is finite. Concurrent mitigation of errors is the main strength of the model and error does not affect the remaining errors. Error removal is all human behavior which is irregular so it cannot be avoid by introducing new errors during the process of error removal.

(MTBF) t =1/ (N-(i-1)) (2) Where, N= Total number of faults. i= Number of fault occurrences. MTBF=Mean Time between failure. t= Time between the occurrence of the $(i-1)^{st}$ and i^{th} fault occurrences.

- G -O Model (GOM): G-O model takes the number of faults per unit time as independent random variables. In this model the number of faults occurred within the time and model estimates the failure time. Delivery of software within cost estimates is also decided by this model.
- Execution Time Model (ETM): Musa's Basic model assumes that all faults are equally likely to occur and Independent of each other. The intensity function is directly proportional to the number of faults remaining in the program and fault correction is proportional to the number of failure occurrence rate. $\mu(t) = \beta 0\ (1-\exp(-\beta 1t))$ (3) Where, $\mu(t)$= mean value function at time t. $\beta 0$=Total number of faults.
- Hyper Exponential Model (HEM): The idea behind this model is that the different parts of the software experience an exponential failure rate. However the rate varies through these parts to ponder different behaviors Different failure rate are placed in different sections $\chi(t)=N\ \Sigma pi\ \beta i(\exp(-\beta i\ t))$ (4) Where, (t)=Failure Intensity Function. t= number of failures. N= finite number of failures.Pi=particular ith class. βi=total number of i^{th} faults.

B. Weibull & Gamma Failure Models (WGFTM)

Models under this category follow per fault failure Gamma distribution instead of exponential distribution.

- Weibull Model (WM): The model is used for hardware reliability. The model incorporates both increasing/decreasing and failure rate due to high flexibility. This model is a finite failure model. MTTF = $\int (1-F(t)) = \int \exp(-\beta t\alpha)$ (5) Where, MTTF = Mean Time to Failure α, β Weibull distribution parameters. t=Time of failure.
- S-Shaped Model (SSM): This model considers that the number of failure with in time period is a Poisson type model .In this model time between failures depends on the time to failure. Mitigation of fault occurs immediately as failure happened.$\mu(t)=\alpha[1-(1+\beta t)e^{-\beta}t]$ (6) Where, (t)=mean value function at time t.

α,β=Distribution parameters. $e^{-\beta}t$= Exponential distribution.

*C. Infinite Failure Time Model (*IFTM*)*

Software is not completely error free when mean value function of a particular model tends to infinity. Models come under the category of Infinite Failure Time Model. Duane's Model (DM): Dunae's observed that an erected line has been generated by

comparing testing time with failure rate .Based on the observations of hardware reliability the same behavior has been observed for software & used for estimating software reliability.
(t)/T =(αT^{β}/)T (7) Where, (t)=Mean value function at time t .
αT^{β} =Expected number of failures by time t. T=Total testing.

- Geometric Model (GM): The model is based on the version of J-M. The time between failures is an exponentially distributed and mean time failure decreased geometrically. E(Xi)=1/Z(ti-1) (8)

Where, E(Xi)= Expected time between failure(ti-1) =Fault detection rate. Logarithmic Poisson (LM): The model assumes that code has an infinite number of failures. The model follows NHPP. When failure occur distribution decreases exponentially The possible number of failures over the time is a logarithmic function therefore it is called
Logarithmic Poisson [19]. χ(t)= χ0 exp(-Ψμ(t)) (9) Where, μ(t)=Mean value function at time t. Ψ=failure rate decay parameter. χ(t)=Failure intensity function.
*D. Bayesian Models(*BM*).* The models are used by several organizations like Motorola, Siemens & Philips for predicting reliability of software .BM incorporates past and current data. Prediction is done on the bases of number of faults that have been found & the amount of failure free operations.
L-V Reliability Growth Model: The model tries to account for fault generation in the fault correction process by allowing for the probability that the software program could become less reliable than before. For every correction of fault, a separate program has to write. Fault correction is obtained by fixing fault
D (i) = μ(1/ χ(i)) (10) Where, χ (i) =Sequence of independent variable. D (i) =Distribution for the i^{th} failure.

III. CONCLUSION

Software is an steady contrivance that encompassed of computer programs, procedures, rules, data and related documentation. The surge in numeral of software fiascos gravely pretentious the recital of conveyance, telecommunication, military, engineering process, showbiz offices, aircrafts and business. Therefore software reliability has become more & more imperative. Reliability is the competence of software to maintain a determined level of recital within the time period. Software reliability is a measuring technique for flaws that causes software failures in which software deeds is different from the specified behavior in a defined environment with static time**.** On the basis of the review the taxonomy on software reliability models has been presented as a major contribution.

REFERENCES

[1] W. K. Ehrlich - Thomas J. Emerson, "Modeling Software

[2] "Critical Review on Software Reliability Models" by A. Yadav1 & R. A. Khan, International Journal of Recent Trends in Engineering, Vol 2, No. 3, November 2009

[3] Failures and reliability Growth during System Testing", AT&T Bell Laboratories, 8 Corporate Place, ACM, 1987, pp. 72-87.

[4] S. Yamada, M. Ohba, and S.Osaki, "S-Shaped Reliability.

[5] Stephen H. Kan - Metrics & Models in Software Quality Engineering.

[6] Growth Modeling for Error Detection," IEEE Tranction Reliability, Dec 1983, pp. 475-478.

[7] A. Wood, "Software Reliability Growth Models", Technical Report 96.1, Tandem Computers, September 1996, pp. 1-30.

[8] J. Lian, "Software Reliability Presentation", ECE-355 Tutorial, 2004, pp. 1-24.

[9] J. M. Caruso, D. W. Dcsormcau, "Integrating Prior Knowledge with a Software Reliability Growth Model", 1BM Corporation, Neighborhood Road, Kingston, NY 12401, IEEE, 1991, pp.238-232.

[10] R.A Khan, K. Mustafa and S.I Ahson, "Operation Profile- A key Factor for Reliability Estimation", Universitiespress, Gautam Das & V P Gulati, CIT, 2004, pp 347-354.

[11] M. R. Lyu, "Software Reliability modeling Survey", IEEE Computer Society Press, AT&T Lab, 1996, pp 71-117.

[12] J. Peter Rooney, "Customer Satisfactions", In Proceedings of annual Reliability and Maintainability Symposium, 24- 27 Jan. 1994 pp.376 – 381.

[13] J. D. Musa and K. Okumoto, "A Logarithmic Poisson Execution Time Model for Software Reliability

[14] Measurement", Bell Lab, IEEE, 1984, pp. 230-238.

[15] M. L. Shooman, "Yes, Software Reliability Can be Measured and Predicted", Division of Computer Science,Polytechnic University, 1987, pp.121-122.

AUTHORS

First Author: Tariq Hussain Sheakh, Lecturer in computer sciences at Govt. Degree College Poonch, J&K,INDIA
Email id - tariqsheakh2000@gmail.com

Second Author: VijayPal Singh, Assistant Professor in Computer Science at JJT University Rajasthan.
Email id - tilotiya09@gmail.com