Volume 3, Issue 8, August 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Comparative Study of Software Reliability Models through Simulation

P.K.Suri[*]
Dean, Research and Development, Chairman CSE/IT/MCA, HCTM Kaithal, Haryana, India

Jagrup
Student, HCTM Kaithal Haryana, India

Abstract— This document gives the reliability computation of the system and comparison between the predicted reliability and the actual reliability of the system. The simulation is done on the data generated by the poisson distribution which describes the nature of the failure process. As the failure in the system can arise randomly at any instant of time, poisson distribution covers this type of process by the definition and nature of the poisson process. Once the required data is generated the reliability of system is calculated by the empirical method and is predicted by the models. The reliability calculations are done on the modified parameters to simulate the effect in variation of model parameters. The aim of variation in model parameter is to get as close as possible to the actual reliability curve for the system. This scheme can be used to predict exact reliability of systems if variations are done in controlled manner and determining the initial parameters with due care and precision. Variations should be kept small with a precision value of 2 to 3 decimal places larger variations affect the reliability prediction with large variations which differ from actual reliability.

Keywords— Software reliability, reliability prediction, reliability estimation, parameter variation, reliability curve, actual reliability

I. Introduction

Increasingly software plays a critical part in every dimension of the industry, education, research and development of new technologies and techniques. Although advances have been made towards the production of defect free software, any software required to operate reliability must still undergo extensive testing and debugging. This can be a costly and time consuming process, and managers require accurate information about how software reliability grows as a result of this process in order to effectively manage their budgets and projects.

According to ANSI, Software Reliability is defined by the probability of failure-free software operation for a specified period of time in a specific environment. Software reliability is an important attribute of software quality [1], together with functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation. Software reliability is hard to calculate for a complex system because the running time of the software may varies from days to years. For such softwares and system it become the primary requirement to study the reliability of the system to avoid major loss of time and money required to develop the entire system.

For such systems the testing phase is very long such system are generally given for beta testing to remove the bugs in system. Before the complete development of such system it is hard to predict the reliability of the system and if the system development is complete at that time it is very costly to remove errors from the system. System reliability prediction[2] is done for such systems because we have many models to predict the software reliability on the bases of the initial parameters observed during the development of the system phases. Reliability prediction if done correctly helps us to predict the actual reliability of the system and is used to adopt different technologies and techniques in the software development. 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.

In reliability prediction the resemblance of actual probability in the predicted reliability is the major concern for the developers. More resemblance with the actual reliability of the predicted reliability [3][4] more helpful it is for a reliable system development. At this time the question arises, how to get a predicted reliability close to actual reliability of the system. Reliability models helps us to predict the reliability of the system. For a reliability model to be effectively utilize , it should be assured about the difference in actual reliability of the system and the predicted reliability by the model. On the bases of knowledge of difference between the actual reliability and predicted reliability another methods can be implemented to find the corrected set of parameters so that our prediction matches the actual reliability calculations. This document is an effort in selection of model parameters for the resemblance of actual reliability with the help of reliability prediction and comparison of effectiveness of different models reliability prediction.

II. Reliability Models

The effects of this process, by which it is hoped software is made more reliable, can be modeled through the use of Software Reliability Growth Models [5],[6],[7] hereafter referred to as SRGMs. Ideally, these models provide a means of characterizing the development process and enable software reliability practitioners to make predictions about the expected future reliability of software under development. Such techniques allow managers to accurately allocate time, money, and human resources to a project, and assess when a piece of software has reached a point where it can be released with some level of confidence in its reliability. Unfortunately, these models are often inaccurate.

Numerous SRGMs have been proposed, and some appear to be better overall than others. Unfortunately, models that are good overall are not always the best choice for a particular data set, and it is not possible to know which model to use *a priori*. Even when an appropriate model is used, the predictions made by a model may still be less accurate than desired. For this reason, a great deal of research has gone into trying to make more effective use of existing models. Various methods have been proposed, such as adjusting for model bias (Brocklehurst, Chan, Littlewood, and Snell 1990[8]; Li and Malaiya 1993; Li 1996), combining multiple models (Lyu and Nikora 1992b)[9], and smoothing of initial data (Li and Malaiya 1993; Li 1996). Li and Malaiya found that the choice of improvement techniques often makes a bigger difference in model accuracy than the initial choice of model used (Li and Malaiya 1993).

Software reliability growth models are commonly designed to be used with data collected in terms of the testing time between failures, and it is on such models that this study will focus. These models are usually stated in terms of two equations; the mean value function and the failure intensity function. It is worth noting that the second function is the derivate of the first. The mean value and failure intensity functions are usually denoted as $\square(t)$and $\square(t)$, respectively. In this case t refers to the total time which has elapsed since the start of testing, as measured in seconds of CPU execution.

A. The Exponential Model

The most widely used software reliability growth model is the exponential model. This is a stochastic model based on a non-homogeneous poisson process (Goel and Okumoto 1979). Originally proposed by Jelinski and Moranda in (Jelinski and Moranda 1971) [10],many variations have since appeared. The original JM-exponential model made use of the elapsed wall clock time when a failure was encountered. A significant refinement was made by Musa, who restated the model in terms of CPU execution time allowing for more accurate predictions.

Musa also described a method for moving between execution time and wall clock time, making it easier to make predictions in terms of real world calendars and deadlines (Musa 1975). Latter, Goel and Okumoto worked to generalize the model, allowing the initial number of errors in a program to be random rather than fixed; and permitting errors to be independent (Goel and Okumoto 1979) [11]. Although superior to the earlier models, it has been shown that the exponential model is not generally the most accurate SRGM (Malaiya, Karunanithi, and Verma 1992). However, this model remains popular and widely used. This study focuses on execution based models, measured in CPU seconds. For this case, the exponential model takes the form

- $\square(t) = \square_0[1-e^{-\square}1^{t}]$ [10]
- $\square(t) = \square_0\square_1 e^{-\square 1t}$

where $\square_0$ is taken to be the total number of defects present in the software, and $\square 1$ is taken to be the per fault hazard rate of the program.

B. The Logarithmic Model

The logarithmic model was originally proposed by Musa and Okumoto in (Musa and Okumoto 1984) [12]. Like the exponential model, it models the failure process as a non-homogeneous Poisson process. The most significant difference between this model and the exponential is that the logarithmic model assumes that failure intensity will decrease exponentially with the expected number of failures experienced, while the exponential model assumes an equal reduction in failure intensity with each fault uncovered and corrected. In this sense it can be viewed as a continuous formulation of the geometric model (Musa and Okumoto 1984).

In (Malaiya, Karunanithi, and Verma 1992) it was shown that the logarithmic model was generally more accurate than many other SRGMs. It is relatively simple to use, although not as widely used as the exponential model. This may be due in part to the difficulty of obtaining a concrete interpretation of the model's parameters.

III. PRESENT WORK

we have compared JM and MO model prediction level and after that we have studied the variation of parameters.

A. Comparison of JM and MO models

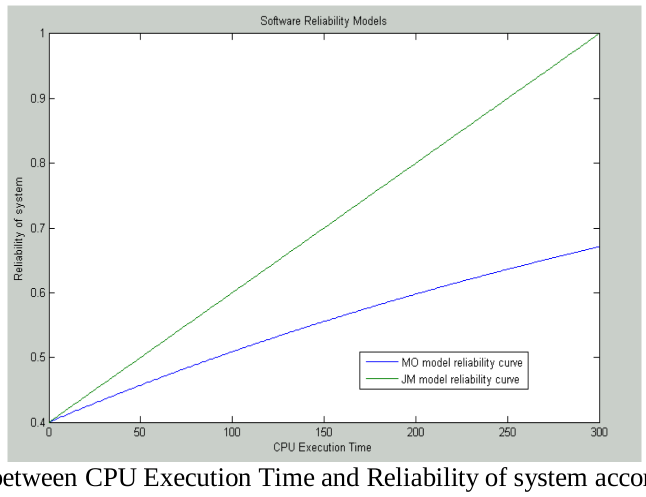

Fig. 1 A line graph drawn between CPU Execution Time and Reliability of system according to two models JM and MO Model.

The two models Jelinski-Moranda(J-M) and Musa- Okumoto (M – O) predicts the reliability of the software on different scales. While we are using the same parameter set for both models there is the difference in the prediction of the reliability of the system.

Musa-Okumoto (M-O) prediction is exponential in nature. While Jelinski–Moranda (J-M) model reliability prediction is different. By this result we want to see how these two models differ in their results.

B. MO Model Reliability Prediction

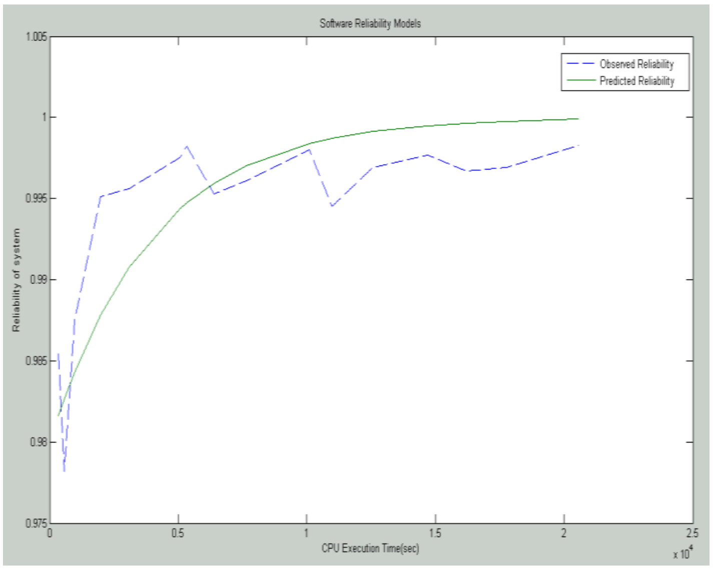

Fig 2: Reliability Curves generated by the Observed Reliability and prediction by MO Model.

The M-O reliability curve is plotted on the bases of the initial parameters while the while the Observed reliability curve is plotted on the bases of actual reliability calculations.

From Fig 2 the two curves we can have an analysis of the predicted reliability and the observed reliability. The variation in two reliability curve can be viewed easily while the prediction curve show the system to reliability equal to 1 at the end of the execution time i.e. development phase , the actual reliability curve shows a small amount of unreliability present in the system.

C. Effect of Variation of parameter on Reliability Prediction in Musa's Basic Model

In Fig 3 we have plotted a set of six curves out of which the first curve that is of the observed reliability is the curve which is based on the actual calculations done on the data set generated in the data generation phase of the simulation. Once the data set is generated and the calculations are done for the reliability curve, we are able to get an estimate of the initial parameter here by known as lemda.

Based on this parameter lemda we predict the reliability of the system which varies slightly from the actual reliability of the software as calculated by actual calculation done.

After this we start varying the value of lemda by increasing its value by a fraction 0.002 and we plot four more curves to find out how close we can go to the actual reliability curve in our prediction by variation in the initial parameter.

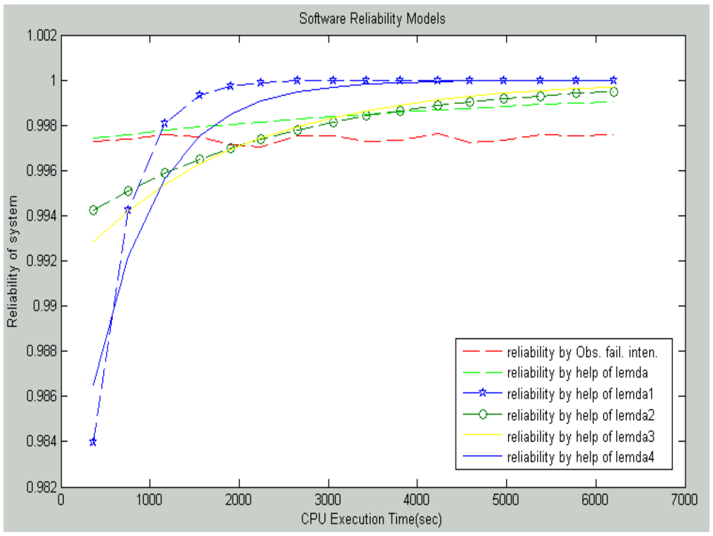

Fig 3: Variation in Parameter and Reliability Prediction in Musa's Basic Model

D. Effect of Variation of parameter on Reliability Prediction in MO Model

Fig 4 plots the simulation results of simulation of the variation of parameter theta used to predict the reliability of the system . Here theta is the original value calculated from the initial value of the observed parameters in the initial phases of the observation while theta1 and theta2 are the result of variations in parameter theta. This simulation result shows that if the parameters are calculated on the bases of actual calculations without any variation than the results returned are more accurate.

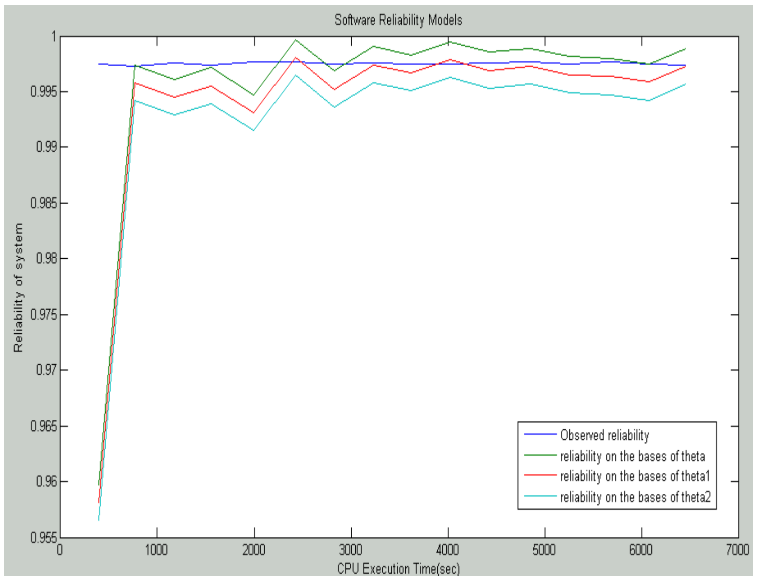

Fig 4: Reliability Curve Comparison with Observed Reliability and Reliability Prediction by MO Model along with Variation in Parameter .

E. Effect of Variation of parameter on Reliability Prediction in JM Model

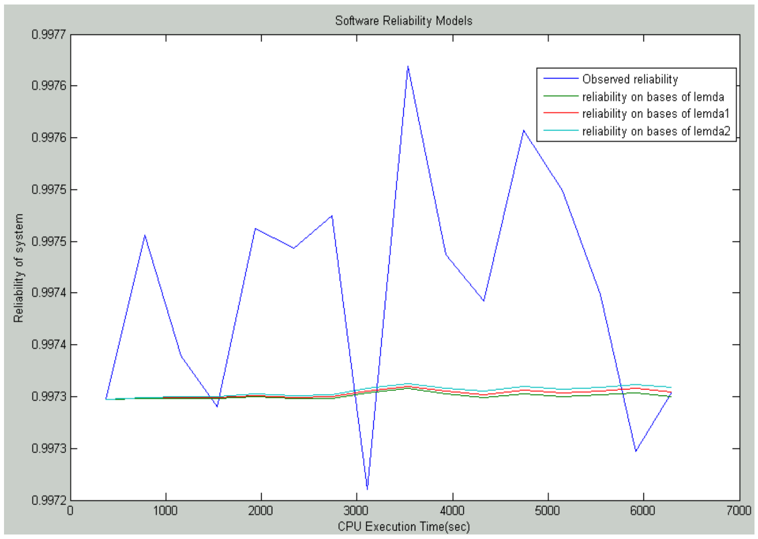

Fig 5: Reliability Curve Comparison with Observed Reliability and Reliability Prediction by JM Model along with Variation in Parameter .

Fig 5 shows the outcome of the variation in initial parameter by a small value and plot of this variation. Aim of each variation is to predict the reliability of the system in the same manner as predicted by the observed parameters and the reliability by the observed parameters. Each variation gives a variation in the predicted reliability which differs in the characteristic. From Fig 5 it is observed that though Observed reliability takes a different path in between initial and final value of reliability but finally it gives the same reliability value as predicted by the reliability model.

IV. Conclusions

Present work examines the models in depth, presenting summaries of previous work done to predict the accuracy. We have examined the issue of how parameters for these models are estimated. Although the existing literature spends a great deal of time in discussing the derivation of the maximum likelihood equations for estimating the parameters of these models, it provides little practical advice to practitioners on which algorithms should be employed to solve these equations. Discussion has suggested that in many cases they have dealt with these issues by linearizing the basic model form and fitting a straight line to the data. We present some recommendations about how to solve the maximum likelihood equations. We compare these two methods not only on data generated from the Poisson distribution. Our results show that variation range does not provide good predictive capabilities, and suffers from a general inability to find valid parameter estimates; for this model the maximum likelihood method is generally superior. When smoothing is applied to the data the least squares method is somewhat more promising, although there is still the problem of inaccurate parameter estimates in many cases.

ACKNOWLEDGMENT

I wish to express my deep sense of gratitude and indebtedness to **Dr. P. K. Suri**, Chairman, Department of Computer Science and Engineering, HCTM, Kaithal for his inspiring guidance, constructive criticism and valuable suggestions during the entire duration of my research work.

I would like to convey my deep regards to all other faculty members and staff of Department of Computer Science And Engineering, HCTM, Kaithal, who have bestowed their great effort and guidance at appropriate times.

REFERENCES

[1] Li, Qiuying; Luo, Lei; Ai, Jun, The Determination Method for Software Reliability Qualitative Indices: 2013 IEEE 7th International Conference on Software Security and Reliability (SERE).2013.

[2] Jintao Zeng, Jinzhong Li, Xiaohui Zeng, Wenlang Luo, A Prototype System of Software Reliability Prediction and Estimation :2010 Third International Symposium on Intelligent Information Technology and Security Informatics(IITSI). Jingangshan, 2010

[3] Wangbong Lee, Boo-Geum Jung, Jongmoon Baik, Early Reliability Prediction: An approach to Software Reliability Assessment in Open Software Adoption Stage: Second International Conference on Secure System Integration and Reliability Improvement,2008. SSIRI'2008.

[4] Bev Littlewood and John L. Verrall, "A Bayesian Reliability Model with a Stochastically Monotone Failure Rate" Verrall Published in IEEE transactions on Reliability,Vol.R-23, No. 2, June 1974.

[5] Swapna S. Gokhale, Michael R. Lyu and Kishor S. Trivedi, "Incorporating Fault Debugging Activities Into Software Reliability Models: A Simulation Approach", at IEEE Transactions On Reliability, Vol. 55, No. 2, June 2006.

[6] Chin-Yu Huang, Sy-Yen Kuo and Michael R.Lyu, "An Assessment of Testing- Effort Dependent Software Reliability Growth Models", in IEEE Transactions on Reliability, Vol. 56, No. 2, pp. 198-211, June 2007.

[7] P. K. Kapur, H. Pham, Sameer Anand, and Kalpana Yadav," Unified Approach For Developing Software Reliability Growth Models ", at IEEE Transactions On Reliability, Vol. 60, No. 1, March 2011.

[8] Brocklehurst, Chan, Littlewood, and Snell , " Recalibrating Software Reliability Models ", at IEEE Transactions on Software Engineering ,1990

[9] M. R. Lyu and A. Nikora, "Applying Reliability Models More Effectively", at IEEE Software , Vol.- 9,No. 4, pp 43-52, July, 1992 .

[10] Michael J. Mangieri, "A Comparison Of Two Software Reliability Engineering (SRE) Models: Jelinski-Moranda And Musa-Okumoto Logarithmic Poisson", At CSMN 658: Software Reliability And Reusability The University Of Maryland University College Graduate School December 7, 2000.

[11] Goel, A.L. and Okumoto, K.," Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures", Published in: Reliability, IEEE Transactions on (Volume:R-28 , Issue: 3) Aug, 1979.

[12] J. D.Musa and K. Okumoto, " A Logarithmic Poisson Execution Time Model For Software Reliability Measurement", in 7th International Conference On Software Engineering, P.230-238, March 26-29, 1984.

[13] Zuzana Krajcuskova, " Software Reliability Models ", at IEEE, 2007.

[14] G. Krishna Mohan, B. Srinivasa Rao and Dr. R. Satya Prasad, " A Comparative Study of Software Reliability Models Using SPC on Ungrouped Data ", published in International Journal of Advanced Research in Computer Science and Software Engineering , Volume 2, Issue 2, February 2012.

[15] P. K. Suri and Neeraj Garg, "Simulator For Evaluating Reliability of Reusable Components In A Domain Interconnection Network", at IJCSNS International Journal of Computer Science and Network Security, Vol. 8 No. 3, March 2008.

[16] Dr. P.K. Suri and Sanjana, "Simulator for Software Project Reliability Estimation ", published in International Journal on Computer Science and Engineering (IJCSE) Vol. 3 No. 7 July 2011.