Volume 7, Issue 5, May 2017 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Software Reliability Modeling: Comparative Analysis

Narender*
Department of Computer Science, Kurukshetra University, Kurukshetra, Haryana, India

Sona Malhotra
Department of Computer Science, Kurukshetra University, Kurukshetra, Haryana, India

Abstract - This paper includes the software reliability prediction and estimation as an area of software development process which needs to specify predict, estimate and asses the software reliability system. Software trustworthiness is an significant part of software quality along with functionality, performance, usability, maintainability, serviceability including many ones. Software reliability related to the software quality and directly affects the system reliability. There are hundreds of models developed from the last thirty to forty years to improve the software trustworthiness. The aim of this piece is to investigation of different software reliability models that can improve the trustworthiness of the software, their classification and finding the best fit for a particular project.

Keywords - Software Reliability Modeling, Neural Network, Fuzzy Logic, Genetic Algorithm, Genetic Programming, Ant Colony, Tabu Search.

I. INTRODUCTION

Reliability of the software we can say is the probability of successful functioning of the software for a particular interval of time under a given environment. Software Reliability is an aspect that greatly affects system consistency. It is different from reliability of hardware system; it imitates the design excellence, rather than manufacturing precision. The high convolution of software is the big contributing aspect of Software related issues. The Reliability of the software is not a function of time - although researchers have moved in the direction of models relating the reliability of the software and time. The modeling practices for steadfastness of the software is reaching its prosperity, but before using any procedure, we must carefully choose the suitable model that can well fit in the particular case. A number of approaches can be used in advancement of the trustworthiness of software; however, it is difficult to balance development time and money with software reliability. Software reliability cannot be directly measured, so the related features are much needed to measure the software reliability. When software is compared these factors are compared among different projects. The factors are:

- Development process
- Fault
- Failure etc.

Complete testing of software is infeasible. Producing defect free software cannot be assured.

II. HISTORY AND BACKGROUND

The chronicle development of the earlier software reliability models have already been presented a couple of times before. Previous reviews of the software reliability modeling have been given by Schick and Wolverton, and Shooman in 1978, 1984 respectively. A book is also published on the earlier models by Musa et al. in 1987. This book showed a greater progress in the area of reliability modeling of the software systems.

Software reliability models have been talked about in sixties but not published at international level. That time was not a hotheaded period for software reliability modeling. There were hardware reliability models that have been used by the end of the sixties. There were some models that were not directly related to software reliability modeling.

III. 1970'S

In 1970's a lot of model discussed and developed to improve the reliability of the software product. Most of them were time between failure model and no. of failure in a particular time interval. They are described in chronicle order according to Software reliability theory [1].

In **1971, Akiyama,** went for a regression analysis of the unsuccessful data where it was noticed that many factors affecting the fault count in a software system. In **1972**, **Jelinski and Moranda,** introduced a model for the assessment of number of early software faults using unsuccessful data. It was considered as first reliability model for the software system. Model assumes that there is finite but unidentified number of existing faults. As the errors are independent, every time when an error is detected, it is deleted without introducing any new error. Failure intensity is relative to the number of residual faults. In the same year, in **1972, Shooman,** also suggested a model that was similar to J-M model where the performance of a key parameter was related to how the project personnel profile varied over time. A number of

mathematical formulas were projected for that. A number of interesting issues related to software development cost problem were discussed. In **1972, Dickson et al.,** published a paper concerned with quantitative analyses of software that no model is produced till date that can exactly quantify the reliability of the software. Though, it is clear from the studyIn **1973, Littlewood and Verrall,** introduced Bayesian software reliability model. The methodology was concerned with estimating the times between successive failures of software. In **1973, Schick and Wolverton,** also suggested two models similar to J-M model. The difference was that they used Rayleigh distribution in modeling time between successive failures. In **1975, Schneidewind,** published a theory for the number of breakdowns discovered during a given time slice and used a non homogenous Poisson process with exponentially diminishing intensity function in modeling software failure process. This appears to be first under NHPP assumption. He suggested investigation of different reliability functions and selecting the best suitable for the particular project. In **1975, Musa,** developed an execution time theory for the assessment of software trustworthiness. Before this, time domain used in software reliability modeling that was entirely calendar time. Using calendar time was resulting in lack of modeling universality. He suggested that execution time is the finest practical measure characterizing error-inducing stress on a program being placed. In **1975, Moranda,** modified the J-M model and published a pair of model one was called de-eutrophication process model and other was geometric Poisson process model. The main benefit was that the time increases between two successive failures in the starting due to the fact that initially detected facts are expected to be greater failure probability. In **1975, Trividi and Shooman,** produced a general Markov model for assessment of software trustworthiness and the software performance state is modeled as upstate and failure state as a down state. In **1978, Wagonar,** studied a same kind of model (Schick and Wolverton, 1973) using a Weibull distribution. In **1978, Lipow,** suggested an amendment (Schick and Wolverton, 1973) by using grouped data in analyzing reliability. He correlated the software metrics with the number of faults to guesstimate and calculate the reliability. In **1979, Goel and Okumotto,** published a simple non homogenous Poisson process model for modeling the failure process in software. That was followed by many researchers later. Many non homogenous models were generalized and modified based on these models that were able to explain software failure process adequately in many cases. In **1979, Littlewood,** produced a model for reliability of the software for modular programs in which transfer of control follows a semi Markov process. It can be looked as a new white box modeling of software trustworthiness. In **1979, Goel and Okumotto,** proposed some software release models. The model also helped in estimating the actual and expected failures.

IV. 1980's

In 1980's no. of models were the modified version of existing model or they were somehow overcoming the existing model. Min Xie [2] well narrated the models in his book Software reliability modeling.

In **1980, Thompson and Chelson,** suggested a bayesian model for calculating the probability of the software is faultless. Both of them were also involved in producing decision rule for carry-over or annihilation of testing. In **1980, Cheung,** produced a customer oriented reliability model which is based on trustworthiness of different models and calculated inter-modular conversion probabilities. In **1981, Moranda,** presented an extension of J-M model with additional constraints and flexibility. In **1985, Musa,** and Okumotto showed that execution time theory is superior over calendar time. In **1987, Musa et al.,** presented the first book on software reliability modeling. That was a great progress of software reliability modeling. In **1987, Schagen and Sallin,** produced a logistic model for eliminating software failure. In **1987, Levendel,** produced a model for fault exposure and fault exclusion. In **1987, Hamlet,** planned a theory of feasible correctness to evaluate the software trustworthiness through testing. In **1987, Mazzuchi and Soyer,** proved that Little and Verral (1979) is an empirical Bayes model. At the same time, in **1987, Xie,** modified J-M model and derived Markov process model by assuming that initially detected fault will contribute more to the failure speed. He wrote a book on earlier models and their amendment for software reliability. He also presented general description of Markov model. In **1988, Wright and Hazelhurst,** amended the J-M model and tried to outpace some difficulties in earlier model. They also focused on modeling of failure rate until the next failure. In **1989, Ohba and Chou,** focused on generalization of some existing model. They considered the possibility of some imperfect debugging. In **1989, Tohma et al.,** proposed a model for residual faults using a hyper geometric distribution. At the same time in **1989, Kubat,** suggested a stochastic model for the conduct of modular software. He used some assumption, and derived system failure rate.

V. 1990's

Study shows that conventional models are based on assumptions in their analytic formulation; as a result these models are not fit in all situations for different projects. A preset behavior where a few factors needed to be adjusted i.e. bending the arc to the unsuccessful data. Their parameters are clearly defined in the model. They have a physical understanding. To regulate the constraints of the parametric models, there are always a set of hypothesis in parametric model that may not be suitable for a number of cases. This kind of models involves conventional models, for example **Jelinski-Moranda (J-M) Model** [3], and the **Geometric Model (GEO)** [4], two most popular conventional models. Non-parametric modeling identifies both the model, and its coefficients. There was an inclusion of parameters in their formulation, but the mentioned parameters do not have a physical interpretation. The above said models are usually based on practices of machine learning techniques, for example: **Artificial Neural Networks (ANN)** [5] [6]. The practices illustrate that ANN non-parametric models show promising results than conventional models. The power of exterior parameters and other oddness of a model can be removed, and the model is capable to transform itself based upon the available unsuccessful data. Though, most of the works look at only time based models. **Genetic Programming (GP)** [7] widens the principles of Genetic Algorithms to ordered search spaces.

Karunanithi et al.[8] was the first person who produced software reliability model based on neural network to predict collective number of breakdowns by using Feed Forward neural networks. He also did some practices by using recurrent neural network. **Sitte**[9] contrasted the approach with rectifying for quantitative measure of parametric models with same datasets using some significant predictive procedures, in the intention of quantifying the reliability of the software. **Cai et al.** [10] proposed the development of a fuzzy model to illustrate software trustworthiness. **Y. Zhang et al.** [11] forecasted for MTBF failure data series of software trustworthiness by genetic programming algorithm.

VI. 2000's

After 1990's most of the model were using the existing model by choosing the reliability function as a fitness function and trained the fitness function using other techniques.

If there is sufficient data, then using of genetic programming permit us to complete the regression of any function practically. Boosting practices have been effectively used to enhance the functioning of other recognized processes from the **Machine Learning field** [12], such as ANN, and GP. **Su et al.** [13] explained the neural network techniques in projects from mathematical point of view of software reliability modeling. **Cai et al.** [14] trained the dataset using back propagation algorithm. They used a number of current 50 failure times as an key in to forecast the next-failure time as output. **Hu et al.** [15] used ANN for the early reliability guess for present projects/releases by reusing the unsuccessful data from earlier releases/projects**. Utkin et al**. [16] presented a software reliability model where times between failures are taken as fuzzy variables ruled by membership functions

Najla Akram AL-Saati et al. [17] guessed parameters based on the existing unsuccessful data. From 1990's there are a majority of reliability models those were using the neural network for training the datasets. The researchers also showed that they were getting the promising results when compared to earlier ones. **Donighi and Mohammadi [18]** defined fuzzy reliability model for software. **Madsen et al. [19]** described the usage of intutionistic fuzzy approach in indefinite software computing.

VII. OBSERVATIONS

On the basis of some fundamental papers and the above technologies the observation concludes:

Table 1. Comparative Analysis

Technology	**Suitable for small datasets**	**Whether redesigned or not for different data sets**	**Applicability for complex situation**
Neural Network	×	×	√
Fuzzy Logic	√	√	√
Genetic Algorithm	Partially	√	√
Genetic Programming	×	×	√
Ant Colony	×	√	√
Tabu Search	Partially	√	√
Cuckoo Search	×	√	√

VIII. CONCLUSION

The study goes through a number of studies and finds that a number of models is not clear whether they are applicable on larger software systems. Most of them rely on assumptions. A number of models are not suitable for all applications. And, above all there is no any software reliability model that can exactly quantify the software. Exact quantification of software reliability is necessary when there is a comparison of reliability of different software, but there is no single model that can exactly quantify the software. Researchers are trying to develop a model that can fulfill the above said issues.

REFERENCES

[1] Michael Rung-Tsong Lyu, "*Software Reliability Theory", Encyclopedia of Software Engineering"* Copyright © 2002 by John Wiley & Sons, Inc. DOI: 10.1002/0471028959.sof 329, 2002.

[2] M.Xie, "*Software Reliability Modeling*", A book published by World Scientific.

[3] P. Moranda and Z. Jelinski, *Final report on software reliability study McDonnall Douglas Astronautics Company*, 1972, Tech. Rep.

[4] P. Moranda, "*Predictions of software reliability during debugging," in Proceedings of the Annual Reliability Maintainability Symposium*, 1975.

[5] S. H. Aljahdali, A. Sheta, and D. Rine, "*Prediction of software reliability: A comparison between regression and neural network non-parametric models*," in CS/IEEE International Conference on Computer Systems and Aplications, 2001, pp. 470–473.

[6] R. Sitte, "*Comparison of software reliability growth predictions: Neural networks vs parametric recalibration,"* IEEE Trans. Software Engineering, vol. 48, no. 3, pp. 285–291, September 1999

[7] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. : MIT Press, 1992.

[8] Karunanithi N., Malaiya Y. K. and Whitley D. ,"*Prediction of software reliability using neural networks*", IEEE, 1991, pp. 124–130.

[9] R. Sitte, "*Comparison of Software Reliability Growth Predictions: Neural networks vs. parametric recalibration*", IEEE transactions on Reliability, pp. 285-291, 1999.

[10] Cai, K.Y., Wen, C.Y., Zhang, M.L., "*A critical review on software reliability modeling*", Reliability Engineering and System Safety 32 (3), 357–371, 1991.

[11] Y. Zhang and H. Chen. *Predicting for MTBF failure data series of software reliability by genetic programming algorithm*. In Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications, Washington, USA, IEEE Computer Society, 2006.

[12] G. Paris, D. Robiliard, and C. Fonlupt, "*Applying boosting techniques to genetic programming,*" in IEEE International Joint Conference on Neural Networks, 2004, pp. 1163–1168.

[13] Yu Shen Su, Chin-Yu Huang, Yi Shin Chen and Jing Xun Chen, "*An Artificial Neural-Network-Based Approach to Software Reliability Assessment*", TENCON, IEEE Region 10, pp-1-6, 2005

[14] K.Y. Cai , L. Cai , W.D. Wang , Z.Y. Yu , D. Zhang, "*On the neural network approach in software reliability modeling",* The Journal of Systems and Software, vol. 58, no. 1, pp. 47-62, 2001.

[15] Q.P. Hu, Y.S. Dai, M. Xie, S.H. Ng, "Early *software reliability prediction with extended ANN model*", Proceedings of the 30th Annual International Computer Software and Applications Conference, pp. 234-239, 2006.

[16] Lev V. Utkin, Sergey V Gurov, Maxim I. Shubinki, *A Fuzzy software reliability model with multiple error introduction and removal, Internation Journal of Reliability Quality Software Engineering*, Volume 09, Issue 3, Page 215, 2002.

[17] Henrik Madsen, Grigore Albeanu, Florin Popentiu Vladiceseu, *An Intutionistic fuzzuy methodology for component based software reliability optimization*, International Journal of Performability Engineering, Volume 8, No. 1, pp. 67-76, 2012.

[18] S, Sardar Donighi, S, Khan Mohammadi, *A fuzzy reliability model for series parallel system*, Internation Journal of Ind. Eng. Vol. 7, No. 12, pp. 10-18, 2011.

[19] Najla Akram AL-Saati and Marwa Abd-AlKareem, "*The Use of Cuckoo Search in Estimating the Parameters of Software Reliability Growth Models"*, (IJCSIS) International Journal of Computer Science and Information Security, Vol. 11, No. 6, 2013.

