On applications of Soft Computing Assisted Analysis for Software Reliability

H.S.Vijaya kumar[1]
Department of MCA
Siddaganga Institute of Technology
Tumkur, Karnataka India
sitvijay@gmail.com

M.A Jayram[2]
Department of MCA
Siddaganga Institute of Technology
Tumkur , Karnataka , India
jayaramdps@gmail.com

Abstract— ***Developing high quality reliable software is one of the main challenges in software industry. Software Reliability is a key concern of many users and developers of software. Demand for software reliability requires robust modeling techniques for software quality prediction. Software reliability models are very useful to estimate the probability of failure of software along with the time. In this study we review the available literature on software reliability. We have also elicited the current trends, existing problems, specific difficulties, future directions and open areas for research.***

I. INTRODUCTION

Software reliability is a key part in software quality along with functionality, usability, performance, serviceability, maintainability and documentation. The standard definition of reliability for software is the probability that a system will continue to function without failure for a specified period in a specified environment. The study of software reliability can be categorized into three parts: modeling, measurement and improvement. Software reliability modeling has matured to the point that meaningful results can be obtained by applying suitable models to the problem. There are many models exist, but no single model can capture a necessary amount of the software characteristics. Assumptions and abstractions must be made to simplify the problem. There is no single model that is universal to all the situations. Software reliability measurement is naïve [1].

Measurement is far from commonplace in software, as in other engineering field. "How good is the software, quantitatively?" As simple as the question is, there is still no good answer. Software reliability can not be directly measured, so other related factors are measured to estimate software reliability and compare it among products. Development process, faults and failures found are all factors related to software reliability. Software reliability improvement is hard. The difficulty of the problem stems from insufficient understanding of software reliability and in general, the characteristics of software [2] . Until now there is no good way to conquer the complexity problem of software. Complete testing of a moderately complex software module is infeasible. Defect-free software product can not be assured. Realistic constraints of time and budget severely limit the effort put into software reliability improvement. If the time and budget is not considered carefully, software reliability can be the reliability bottleneck of the whole system. Ensuring software reliability is no easy task. As hard as the problem is, promising progresses are still being made toward more reliable software. More standard components and better process are introduced in software engineering field. In the last few years many research studies has been carried out in this area of software reliability modeling and forecasting. They included the application of fuzzy logic models neural networks, Genetic algorithms (GA) based neural networks, Recurrent neural networks, Bayesian neural networks, and support vector machine (SVM) based techniques, to name a few.

Software Reliability issues are focused on Soft computing techniques for developing and maintaining software systems whose reliability can be quantitatively evaluated. In order to estimate as well as to predict the reliability of software systems, failure data need to be properly measured by various means during software development and operational phases. Although software reliability has remained an active research subject over the past forty years, challenges and open questions still exist. The various modeling technique for software reliability is reaching its prosperity, but before using these techniques, we must carefully select the appropriate soft computing tool that can best suit the case in question. Measurement in software is still in its infancy. No good quantitative methods seem to have been developed to represent software reliability without excessive limitations.

II. Software Reliability Prediction Using Fuzzy Logic

Fuzzy logic is derived from fuzzy set theory which deals with reasoning which is approximate rather than precisely deduced from classical predicate logic. It can be thought of as the application side of fuzzy set theory dealing with well thought out real world expert values for a complex problem.[3] Fuzzy logic was initiated in 1965 [4][5] by Lotfi A.Zadeh, professor of computer science. Fuzzy logic is proven to be capable of modeling highly nonlinear and multidimensional models. Fuzziness refers to non statistical imprecision and vagueness in information and data. The difference between fuzzy logic and probabilistic logic consists in the fact that the fuzzy logic uses truth degrees as a mathematical model for vague facts while the probabilistic one is a mathematical model for random facts [6]. The linguistic values are used for writing the If – Then rules. Researchers in this area have felt that fuzzy logic is vital for Software reliability prediction. Yuan et. Al in [7] used fuzzy subtractive clustering integrated with module order modeling for software quality prediction. First Fuzzy Subtractive clustering is used to predict the number of faults then module order modeling is used to predict whether modules are fault prone or not. Xu et al [8] introduced the fuzzy nonlinear regression (FNR) modeling technique as a method for predicting fault ranges in software modules. A case study of full scale industrial software system was used to illustrate the usefulness of FNR Modeling. Jeff Tian in [9] assessed software reliability by grouping data into clusters. The series of data clusters associated with different time segments are used directly as a piecewise linear model for reliability assessment and problem identification. The model is evaluated in the testing of two large software systems from IBM. Adnan et al in [10] explored the potential of prediction techniques which have been used for assessing software reliability. The DACS Services at the Department of Defense (DoD) Software Information clearinghouse provides an authoritative source for the state of the art software information, supplying technical support for the software community. John Musa of Bell Telephone Laboratories compiled a software reliability database.. The dataset consists of software failure data on 16 projects. Careful controls were employed during data collection to ensure that the data would be of high quality. The data was collected throughout the mid 1970s. It represents projects from a variety of applications including real time command and control, word processing, commercial, and military applications. Cai et al. [11] advocated the development of fuzzy software reliability models in place of probabilistic software reliability models (PSRMs). Their argument was based on the proof that software reliability is fuzzy in nature. A demonstration of how to develop a fuzzy model to characterize software reliability was also presented. Karunanithi et al. [12] carried out a detailed study to explain the use of connectionist models in software reliability growth prediction. It was shown through empirical results that the connectionist models adapt well across different datasets and exhibit better predictive accuracy than the well-known analytical software reliability growth models.

III. Software Reliability Prediction Using Artificial Neural Networks

In recent years, many papers have presented in various models for software reliability prediction. In this section, some works related to Artificial neural network modeling for software reliability modeling and prediction is presented. Many factors like software development process, and software test or use characteristics, software complexity, and nature of software faults and the possibility of occurrence of failure affect the software reliability behavior. Neural network methods normally approximate any non linear continuous function. So more attention is given to neural network based methods now-a-days. Neural network based software reliability model was first presented by Karunanithi et al. [13][14] to predict cumulative number of failures. They consider execution time as the input of the neural network. In their approach they used different networks like Feed Forward neural networks, Recurrent neural networks like Jordan neural network and Elman neural network. Two different training regimes like Prediction and Generalization are also used in their study. They compared their results with some statistical models and found better prediction than those models. Karunanithi et al. [15] also used connectionist models for software reliability prediction. They applied the Falman's cascade Correlation algorithm to find out the architecture of the neural network. They considered the minimum number of training points as three and calculated the average error (AE) for both end point and next-step prediction. Their results concluded that the connectionist approach is better for end point prediction.

Cai et al. [16] Proposed a neural network based method for software reliability prediction. They used back propagation algorithm for training. They evaluated the performance of the approach by varying the number of input nodes and number of hidden nodes. They concluded that the effectiveness of the approach generally depends upon the nature of the handled data sets. Tian and Noore [17] proposed an on-line adaptive software reliability prediction model using evolutionary connectionist approach based on multiple-delayed-input single-output architecture. The proposed approach, as shown by their results, had a better performance with respect to next-step predictability compared to existing neural network model for failure time prediction. Tian and Noore [18] proposed an evolutionary neural network modeling approach for software cumulative failure time prediction. Their results were found to be better than the existing neural network models. It was also shown that the neural network architecture has a great impact on the performance of the network. Viswanath [19] proposed two models such as neural network based exponential encoding and neural network based logarithmic encoding for prediction of cumulative number of failures in software. He encoded the input i.e. the execution time using the above two encoding scheme. He applied the approach on four datasets and compared the result of the approach with some statistical models and found better result than those models.

Ho et al. [20] performed a comprehensive study of connectionist models and their applicability to software reliability prediction and found them to be better and more flexible than the traditional models. A comparative study was performed between their proposed modified Elman recurrent neural network, with the more popular feed forward neural network, the Jordan recurrent model, and some traditional software reliability growth models. Numerical results show that the proposed network architecture performed better than the other models in terms of predictions. Despite of the recent advancements in the software reliability growth models, it was observed that different models have different predictive capabilities and also no single model is suitable under all circumstances. Pai and Hong [21] have applied support vector machines (SVMs) for forecasting software reliability where simulated annealing (SA) algorithm was used to select the parameters of the SVM model. The experimental results show that the proposed model gave better predictions than the other compared methods. Su and Huang [22] showed how to apply neural networks to predict software reliability. Further they made use of the neural network approach to build a dynamic weighted combinational model (DWCM) and experimental results show that the proposed model gave significantly better predictions Use of ANN for predicting the defects before taking up a project implementation well before the beginning the project. Aljahdali et al. [23], made contributions to software reliability growth prediction using neural networks by predicting accumulated faults in a determined time interval. They use a feed forward neural network in which the number of neurons in the input layer represents the number of delay in the input data. All of the above mentioned models only consider single neural network for software reliability prediction. In [24][25], it was presented that the performance of a neural network system can be significantly improved by combining a number of neural networks. Jheng [27] presented neural network ensembles for software reliability prediction. He applied the approach on two software data sets and compared the result with single neural network model and some statistical models. Experimental results show that neural network ensembles have better predictive capability. With the help of the feed forward neural network for software reliability prediction Sing et al [27][28] applied back propagation algorithm to predict software reliability growth trend The experimental result had shown that the proposed system has better prediction than some traditional software reliability growth models.

IV. Software Reliability Prediction Using Genetic Algorithm

Genetic Algorithms (GAs) were developed by Prof. John Holland and his students at the University of Michigan during the 1960s and 1970s. Genetic algorithm can be used by represent a solution to your problem as a genome (or chromosome) [5]. The genetic algorithm then creates a population of solutions and applies genetic operators such as mutation and crossover to evolve the solutions in order to find the best one. The three most important aspects of using genetic algorithms are: (1) definition of the objective function, (2) definition and implementation of the genetic representation, and (3) definition and implementation of the genetic operators. Once these three have been defined, the generic genetic algorithm should work fairly well. Beyond that you can try many different variations to improve performance, find multiple optima (species - if they exist), or parallelize the algorithms [10].Genetic algorithms are machine learning and optimization schemes, much like neural networks. However, genetic algorithms do not appear to suffer from local minima as badly as neural networks do. Genetic algorithms are based on the model of evolution, in which a population evolves towards overall fitness, even though individuals perish. Evolution dictates that superior individuals have a better chance of reproducing than inferior individuals, and thus are more likely to pass their superior traits on to the next generation. This “survival of the fittest” criterion was first converted to an optimization algorithm by Holland in 1975 ,and is today a major optimization technique for complex, nonlinear problems .Oliveira et al. [18, 19] proposed the using of genetic programming (GP) to obtain software reliability model for forecasting the reliability and extended this work by boosting the GP algorithm using re-weighting. The re-weighting algorithm calls many times the learning algorithm with assigned weights to each example. Each time, the weights are computed according to the error (or loss) on each example in the learning algorithm. In this way, the learning algorithm is manipulated to look closer at examples with bad prediction functions. Sheta [16] uses genetic algorithms to estimate the COCOMO model parameters for NASA Software Projects. The same idea is implemented for estimating the parameters of different SRGM models using PSO [17].

V. Issues and challenges

In order to estimate as well as to predict the reliability of software systems, failure data need to be properly measured by various means during software development and operational phases. Although software reliability has remained an active research subject over the past 35 years, challenges and open questions still exist. The various modeling technique for Software Reliability is reaching its prosperity, but before using these technique, we must carefully select the appropriate model that can best suit our case. Measurement in software is still in its infancy. No good quantitative methods have been developed to represent Software Reliability without excessive limitations.

VI. Future Directions:

Software Reliability Engineering relates to whole software life cycle. We discuss possible future directions with respect to four areas: software architecture, testing and metrics [1].

A. Reliability for software architectures and off-the- shelf components

Due to the ever-increasing complexity of software systems, modern software is seldom built from scratch. Revolutionary and evolutionary object-oriented design and programming paradigms have vigorously pushed software reuse. In the light of this shift, reliability engineering for software development is focusing on two major aspects: software architecture, and component-based software engineering. The software architecture of a system consists of software components, their external properties, and their relationships with one another. As software architecture is the foundation of the final software product, the design and management of software architecture is becoming the dominant factor in software reliability engineering research. In this popular software development technique, many research issues are identified, such as reliability, software reusability, clean interface design, fault tolerance etc.

B. Testing for reliability assessment

Software testing and software reliability have traditionally belonged to two separate communities. Software testers test software without referring to how software will operate in the field, as often the environment cannot be fully represented in the laboratory. Software reliability measurers insist that software should be tested according to its operational profile in order to allow accurate reliability estimation and prediction. One approach is to measure the test compression factor, which is defined as the ratio between the mean time between failures during operation and during testing. Another approach is to ascertain how other testing related factors can be incorporated into software reliability modeling, so that accurate measures can be obtained based on the effectiveness of testing efforts.

C. Metrics for reliability prediction

Today companies must collect software metrics as an indication of a maturing software development process. Industrial software engineering data, particularly those related to system failures, are historically hard to obtain across a range of organizations. Novel methods are used to improve reliability prediction are actively being researched. For example, by extracting rich information from metrics data using a sound statistical and probability foundation, Moreover, traditional reliability models can be enhanced to incorporate some testing completeness or effectiveness metrics, such as code coverage, as well as their traditional testing-time based metrics. The key idea is that failure detection is not only related to the time that the software is under testing, but also what fraction of the code has been executed by the testing.

Conclusions

This paper presented the usages of the soft computing techniques for software reliability. Employing effective software reliability engineering techniques to improve product and process reliability would be the industry's best interests as well as major challenges. As the majority of faults are found in a few of its modules so there is a need to investigate the modules that are affected severely as compared to other modules and proper maintenance need to be done in time especially for the critical applications. Preliminary results are quite interesting and more insights will provide a special soft computing assisted architecture for enabling the specialist in software reliability engineering.

References:

[1] M R.Lyu, Handbook of software reliability engineering, New York: McGraw-Hill, 1996.

[2] S. Amasaki, O. Mizuno, T. Kikuno, and Y. Takagi, "A Bayesian Belief Network for Predicting Residual Faults in Software Products," Proceedings of 14th International Symposium on Software Reliability Engineering (ISSRE2003), November 2003.

[3] Klir, George J., St Clair, Ute H., Yuan, "Fuzzy set theory: foundations and applications".

[4] L.A. Zadeh,"Fuzzy algorithms, Info. & Ctl.", Vol. 12, 1968.

[5] L.A. Zadeh, "Fuzzy Sets, Information and Control", 1965.

[6] N. Constantinescu, I. Iancu, Fuzzy Identity Authentication, Latest Trends on Computers, 1(1), pp 168-173, 2008.

[7]Tian, J, " Better Reliability Assessment an Prediction through Data Clustering", IEEE Transactions on Software Engineering, October 2002, Vol. 28, No. 10.

[8] Xu, Z, Allen, E, B, "Prediction of Software Faults Using Fuzzy Nonlinear Regression Modeling", IEEE Software, 2000, 281-290.

[9]. Yuan, X, Ganesan, K, "An application of Fuzzy clustering to software quality prediction", IEEE Software, 2000

[10]. Adnan, W,A, Mashkuri, H,Y, "An Integrated Neural Fuzzy System of Software Reliability Prediction", IEEE Software.

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

[12]. N., Karunanithi, D., Whitley, and Y., Maliya, Prediction of software reliability using connectionist models. IEEE Transactions on Software Engineering 18, 563–574, 1992

[13] N. Karunanithi , Y.K. Malaiya , D. Whitley, Prediction of software reliability using neural networks, Proceedings of the Second IEEE International Symposium on Software Reliability Engineering, pp. 124–130, 1991.

[14] N.Karunanithi, D. Whitley, Y.K. Malaiya, Prediction of software reliability using connectionist models, IEEE Trans Software Engg., vol. 18, no. 7, pp. 563-573, 1992.

[15] N. Karunanithi, D. Whitley, Y.K. Malaiya, Using neural networks in reliability prediction, IEEE Software, vol. 9, no. 4, pp.53–59, 1992.

[16] 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.

[17]. L., Tian, and A., Noore, Evolutionary neural network modeling for software cumulative failure time prediction. Reliability Engineering and System Safety 87, 45–51, 2005.

[18]. L., Tian, and A., Noore, On-line prediction of software reliability using an evolutionary connectionist model. The Journal of Systems and Software 77, 173–180, 2005.

[19] S.P.K Viswanath, Software Reliability Prediction using Neural Networks, PhD. Thesis, Indian Institute of Technology Kharagpur,.

[20].S.L., Ho, M., Xie, and T.N., Goh, A study of connectionist models for software reliability prediction. Computers and Mathematics with Applications 46 (7), 1037–1045, 2003.

[21]. P. F., Pai, and W.C., Hong, Software reliability forecasting by support vector machines with simulated vector machines with simulated annealing algorithms. The Journal of Systems and Software 79, 747-755, 2006

[22]. Y., Su, and C.,Huang, Neural Network-Based Approaches for Software Reliability Estimation using Dynamic Weighted Combinational Models. Journal of Systems and Software 80 (4), 606–615, 2006.

[23].S., Aljahdali A., Sheta and R., Rine, Predicting accumulated faults in software testing process using radial basis function network models. In 17th International Conference on Computers and Their

