International Journal of Research and Development in Applied Science and Engineering (IJRDASE)
ISSN: 2454-6844

Software Reliability and Quality Assessment Models – A Review of Literature

Khushter Kaifi
Department of CSE
Shri Venkateshwara University
k.kaifi@hotmail.com

Dr. Anshu Srivastava
Department of CSE
Shri Venkateshwara University
anshuqrat114@gmail.com

Dr. Qamar Parvez Rana
Dept of Computer Center
Jamia Hamdard University
qprana@jamaihamdard.ac.in

***Abstract*--Size, complexity, and human dependency on software-based products have grown dramatically during past decades. Software reliability assessment is a very vital factor to characterise the quality of any software product quantitatively during testing phase. Software reliability and quality prediction is highly desired by the stakeholders, developers, managers, and end users. Detecting software faults early during development will definitely improve the reliability and quality in cost-effective way. a review of the recent available literature on software reliability and quality prediction is presented. The following sections cover the literature surveys on software reliability models, reliability-relevant software metrics, software capability maturity models, software defect prediction model, and software quality prediction models, regression testing and test case prioritization, and operational profile-based testing.**

Key Words: Software Reliability Assessment, Software Faults, Software Quality, Regression Testing

1 Introduction

Size, complexity, and human dependency on software-based products have grown dramatically during past decades [1]. Software developers are struggling to deliver reliable software with acceptable level of quality, within given budget and schedule. The probability that a software will perform a required function under stated conditions for a specified period of time is known as software reliability. Software reliability assessment is a very vital factor to characterise the quality of any software product quantitatively during testing phase [1].

One measure of software quality and reliability is the number of residual faults. Therefore, researchers are focusing on the identification of the number of fault presents in the software or identification of program modules that are most likely to contain faults. A lot of models have been developed using various techniques. A common approach is followed for software reliability prediction utilizing failure data. Software reliability and quality prediction is highly desired by the stakeholders, developers, managers, and end users. Detecting software faults early during development will definitely improve the reliability and quality in cost-effective way [1].

A software is said to contain a fault if, for input data, the output result is incorrect [2]. A fault is always an existing part in software codes. Therefore, the process of software debugging is a fundamental task of the life cycle of a software system. During this period, the software program is tested many times with the intent of discovering faults contained. When a failure is observed, the code is inspected to find the fault which caused the software failure. The fault is usually removed by correcting the software codes. As a result, one expects the software reliability to increase during the testing phase as more and more faults are removed. The reliability improvement phenomenon is called reliability growth. The size and the complexity of the software packages make it impossible to find and correct all existing faults. The best thing is to give software a reliability requirement and to try to attain a goal by testing the software and correcting the detected faults. However, obtaining the required software reliability is not an easy task. Thus, high reliability is usually estimated by using appropriate models applied on failure data from the software failure history.

In this paper, a review of the recent available literature on software reliability and quality prediction is presented. The following sections cover the literature surveys on software reliability models, reliability-relevant software metrics, software capability maturity models, software defect prediction model, and software quality prediction models, regression testing and test case prioritization, and operational profile-based testing.

2 A Review of Various Models

2.1 Software Reliability Models

A number of analytical models have been presented in literature to address the problem of software reliability measurement. These approaches are based mainly on the failure history of software and can be classified according to the nature of the failure process. The various software reliability models may be categorized as failure rate model, fault count model, software reliability growth models, etc. Details about these models can be found in Musa et al. (1987), Goel and Okumoto (1979), Pham (2006), Lyu (1996). These reliability prediction models attempt to predict the reliability of the software in the later stages of the life cycle (testing and beyond). However, the opportunity of controlling software development process for cost-effectiveness is missed.

Software reliability models usually refer to estimating the number of remaining errors in partially debugged software. Tests performed on the software derive outcome as accepted, conditionally accepted, or rejected. Acceptance may be based on the number of errors found over a selected period of time, on the number of paths executed of the total number of paths available to be executed, or some other

[1] ranged criterion. Variety of reliability models are now competing for the attention of the analyst.

Software reliability models can be broadly categorized as suggested by [3]:

• Deterministic used to study the number of distinct operators and operands as well as the machine instructions in the program. Two most common deterministic models are:

– Halstead's software metric, based on unique no. of operators and operands.

– McCabe's cyclomatic complexity metric, based on cyclomatic number V(G).

• Probabilistic describes the failure occurrence and\or fault removal phenomenon of the testing process as probabilistic events with respect to time and\or testing effort. Some common probabilistic models include the following [3]:

– Failure rate model (times between failure models).

– Failure or fault count model (NHPP models).

– Error or fault-seeding model.

– Reliability growth model, etc.

2.2 Architecture-based Software Reliability Models [7][1]

The software reliability prediction models which are based on number of fault/failures and times between failures are called black-box approach to predict software reliability. The common feature of black-box models is the stochastic modeling of the failure process, assuming some parametric model of cumulative number of failures over a finite time interval or of the time between failures [1]. The main goal of architecture-based software reliability model is to estimate the system reliability by considering the architecture of the software components and their interaction with other ones.

2.3 Bayesian Models [1][8]

This group of models views reliability growth and prediction in a Bayesian framework rather than simply counting number of faults or failures. Failure and fault count model assumes that impact of each fault will be the same with respect to reliability.

2.4 Early Software Reliability Prediction Models [1]

All the approaches discussed earlier for reliability prediction attempt to predict the reliability of the software in the later stages of the life cycle (testing and beyond). However, the opportunity of controlling software development process for cost effectiveness is missed. Therefore, the need of early software reliability prediction is realized. Early reliability prediction attracts software professionals as it provides an opportunity for the early identification of software quality, cost overrun, and optimal development strategies. During the requirements, design, or coding phase, predicting the number of faults can lead to mitigating actions such as additional reviews and more extensive testing.

2.5 Reliability-Relevant Software Metrics

In order to achieve high software reliability, the number of faults in delivered code should be reduced. Furthermore, to achieve the target software reliability efficiently and effectively, it needs to be known at early stages of software development process. One way of knowing software reliability during early stages of development is early software reliability prediction. Since early phase of software life cycle testing/field failure data is not available, information available such as reliability-relevant software metrics, developer's maturity level, and expert opinions can be utilized to predict the number of faults in the software.

2.6 Software Capability Maturity Models [1][12]

The capability maturity model (CMM) has become a popular methodology to develop high-quality software within budget and time. The CMM framework includes 18 key process areas such as quality assurance, configuration management, defect prevention, peer review, and training. A software process is assigned the highest maturity level if the practices in all 18 key process areas of the CMM are adopted. The CMM practices aid in reducing defect injection and in early identification of defects. As a consequence, the number of errors detected in testing and remaining in the delivered software will become lesser. [13] provided an overview of the Capability Maturity Model for software development process.

2.7 Software Defects Prediction Models

Reliability of software system can be adversely affected by the number of residual faults present in the system. The main goal of software developers is to minimize the number of faults (defects) in the delivered code. An objective of all software projects is to minimize the number of residual faults in the delivered code and thus improving quality and reliability. Improving reliability is a key objective during system development and field deployment, and defect removal is the bottleneck in achieving this objective.

2.8 Software Quality Prediction Models

Assuring quality of large and complex software systems are challenging as these systems are developed by integrating various independent software modules. The quality of the system will depend on the quality of individual modules. All the modules are neither equally important nor do they contain an equal amount of faults. Therefore, researchers started focusing on the classification of these modules as fault-prone and not fault-prone. Software reliability and quality prediction model is of a great interest among the software quality researchers and industry professionals. Software quality prediction can be done by predicting the expected number of software faults in the modules or classifying the software module as fault-prone (FP) or not fault-prone (NFP). A commonly used software quality classification model is to classify the software modules into one of the following two categories: FP and NFP. A lot of efforts have been made for FP module prediction using various methods such as classification tree, neural networks, support vector machine, fuzzy logic and logistic regression.

Various classification models have been developed for classifying a software module as FP and NFP. [14] utilizes logistic regression in combination with Boolean discriminant functions for predicting FP software modules. [15] incorporated three different regression tree algorithms CART-LS, S-PLUS, and CART-LAD into a single case study to show their effectiveness in finding the number of faults predicted using them. A study conducted by [16]

compared the fault prediction accuracies of six commonly used prediction modeling techniques. From the literature, it has been found that the decision tree induction algorithms such as CART, ID3, and C4.5 are efficient techniques for FP module classification. These algorithms uses crisp value of software metrics and classify the module as a FP or NFP.

3. Review of Literature

Below is given a brief review of the work done by many workers in the above field of defect prediction with the objective of finding out future strategies in this field.

Bibi S., Tsoumakas G., Stamelos I., Vlahavas I.(2006) used machine learning technique for finding out the number of defects viz. called Regression via Classification (RvC). A comparative experimental study of many machine learning algorithms was carried out in order to evaluate this approach.

Norman Fenton et.al.(1999), have described a probabilistic model for software defect prediction. The aim here is to design a model which is a combination of diverse forms that may be often casual, with available evidence in development of software so that the work can be done in more natural and efficient manner than it was previously done. Here a *critical* review of numerous software metrics and statistical models and the state-of-the art has been carried out.

Ahmet Okutan, et.al.(2012), proposed a novel method using Bayesian networks to explore the relationships among software metrics and defect proneness. Nine data sets from Promise data repository has been used and show that RFC, LOC, and LOCQ are more effective on defect proneness. Also proposal for two more metrics, i.e. NOD for the number of developers and LOCQ for the source code quality has been given.

Mrinal Singh Rawat et. al.(2012), identified causative factors which in turn suggest the remedies to improve software quality and productivity. They showed how the various defect prediction models are implemented resulting in reduced magnitude of defects. They presented the use of various machine learning techniques for the software fault prediction problem.

Supreet Kaur, et.al. (2012), did performance analysis of Density-Based Spatial Clustering of Applications with Noise. It was used for error forecasting in OOSS and C++ language based software parts.

Xiao-dong Mu et. al.,(2012), in their work to improve the accuracy of software defect prediction, a co-evolutionary algorithm based on the competitive organization is put forward for software defect prediction. During this algorithm, firstly, competition mechanism is introduced to organization coevolutionary algorithm. Then, three evolution operators which are reduced operator, allied operators and disturbed operators are developed for evolution of population. And competition is considered for calculate the fitness function. When the algorithm applied into software defect prediction, it improves the accuracy of software prediction through increases the diversity of population.

N Fenton, et. al. (2008), used Baysian networks for forecasting of reliability and defectiveness of software. It used casual process factors and qualitative and quantitative measure, thus catering to the limitations of traditional software limitations.

Jie Xu, et. al. (2010), various statistical techniques and machine learning methods were used to variefy the validity of software defect prediction models..

Manu Banga, (2013), here a new computational intelligence sequential hybrid architectures involving Genetic Programming (GP) and Group Method of Data Handling (GMDH) viz. GPGMDH have been discussed.

Mohamad Mahdi Askari and Vahid Khatibi Bardsiri (2014) for the prediction of software defects used artificial neural network in order to better the generalization capability of the algorithm. Further support vector machine technique was used along with the learning algorithm and evolutionary technique.

Kamaljit Kaur (2012) led to the identification of reusable software modules in OOPS using neural network technique. Used metrics for structural analysis.

Mrs.Agasta Adline, Ramachandran. M(2014) attempted to predict the fault–proneness of a program modules when fault labels for modules are not present. Supervised techniques like Genetic algorithm based software fault prediction approach for classification has been proposed.

Karpagavadivu.K, et.al. (2012) analyzed the performance of various techniques used in software fault prediction. Also described some algorithms and its uses. They found that the aim of the fault prone module prediction using data mining is to improve the quality of software development process.

Xiaoxing Yang, et.al. (2014) used the rank performance optimization technique for software forecasting model development. For this rank to learning approach was used. The model was developed on previous work and was later studied for improving the performance of the model. . The work includes two aspects: one is a novel application of the learning-to-rank approach to real-world data sets for software defect prediction, and the other is a comprehensive evaluation and comparison of the learning-to-rank method against other algorithms that have been used for predicting the order of software modules according to the predicted number of defects. This study shows that the effect of optimization of the model performance using rank to learning approach truly improve the prediction accuracy.

Ahmet Okutan and Olcay Taner Yıldız, (2013) proposed a new kernel method to predict the number of defects in the software modules (classes or files). The proposed method is based on a pre-computed kernel matrix which is based on the similarities among the modules of the software system. Novel kernel method with existing kernels in the literature (linear and RBF kernels) has been compared and show that it achieves comparable results.

Yajnaseni Dash, Sanjay Kumar Dubey, (2012) surveyed different research methods for the prediction of OO metric using neural network techniques. This technique was found to be best suited for prediction in case of object oriented metrics.

Ms. Puneet Jai Kaur, Ms. Pallavi, (2013) used different data mining techniques for software error prediction, like association mining, classification and clustering techniques.

Sonali Agarwal and Divya Tomar, (2014) proposed a feature selection based Linear Twin Support Vector Machine (LSTSVM) model to predict defect prone software

features. F-score, a feature selection technique, is used to determine the significant metrics set which prominently affects the defect prediction in a software module.

Romi Satria Wahono and Nanna Suryana (2013) proposed the combination of particle swarm optimization and bagging technique for improving the accuracy of the software defect prediction.

K.Venkata Subba Reddy and Dr.B.Raveendra Babu (2013) proposed a software reliability growth model, which relatively early in the testing and debugging phase, provides accurate parameters estimation, gives a very good failure behaviour prediction and enable software developers to predict when to conclude testing, release the software and avoid over testing in order to cut the cost during the development and the maintenance of the software. Two real world experimental data previously analyzed have been used to compare our proposed Early Estimation Logistic Model effectiveness with several pre-existing models.

4. Observations

On reviewing literatures, the following observations have emerged:

1. Failure data are not available in the early phases of software life cycle and the information such as reliability-relevant software metrics, developer's maturity level, and expert opinions can be used. Both software metrics and process maturity play a vital role in early fault prediction in the absence of failure data. Therefore, integrating software metrics with process maturity will provide a
better fault prediction accuracy.

2. Early fault prediction is useful for both software engineers and managers since it provides vital information for making design and resource allocation decisions and thereby facilitates efficient and effective development process. Therefore, a model to predict number of faults present at the end of each phase of software life cycle is required. An early fault prediction model using software metrics has been discussed in chapter 4 and 5.

3. One of the measures of software reliability is the number of residual faults and system reliability will be lesser as the number of residual defects (faults) in the system becomes more. There are several faults prediction models, but predicting faults without field failure data before testing are rarely discussed. Therefore, there is a need of a fault prediction model which predicts number of residual faults which may likely to occur after testing or operational use.

4. Prediction of exact number of fault in a software module is not always necessary and there must be some measure of categorization which classify software module as FP or NFP. This will definitely help to improve the reliability and quality of software products by better resource utilization during software development process.

5. Regression testing is vital for reliability assurance of a modified program. It is one of the most expensive testings.

REFERENCES

[1] A. K. Pandey and N. K. Goyal, (2013), "Early Software Reliability Prediction, Studies in Fuzziness and Soft Computing", Springer India, PP. 17-33.

[2] Abdelelah M. Mostafa, (2006), "Regression approach to software reliability models", Graduate Theses and Dissertations, University of Soouth Florida.

[3] Pham, H. (2006). System software reliability, reliability engineering series. London: Springer.

[4] Lyu, M. R. (1996). Handbook of software reliability engineering. NY: McGraw–Hill/IEE, Computer Society Press.

[5] Goel, A. L. (1985). Software reliability models: assumptions, limitations, and applicability. IEEE Transaction on Software Engineering, SE–11(12), 1411–1423.

[6] Shailee Lohmor, Dr. B B Sagar, (2014), "Overview: Software Reliability Growth Models, International Journal of Computer Science and Information Technologies, Vol. 5 (4) , 5545-5547.

[7] Swapna S. Gokhale, (2007), "Architecture-Based Software Reliability Analysis: Overview and Limitations, IEEE Transactions On Dependable And Secure Computing, Vol. 4, NO. 1,pp. 32-40.

[8] N Fenton, M Neil, and D Marquez, (2008), "Using Bayesian networks to predict software defects and reliability", Proc. I MechE Vol. 222 Part O: J. Risk and Reliability, pp. 701-712.

[9] John G. Leonard, et. al. (1996), "An Analysis Of Early Software Reliability Improvement Techniques", Thesis. https://**standard**s.**ieee**.org/find**stds**/**standard**/**982.2**-1988.html

[10] Norman Fenton, Paul Krause and Martin Neil, (1999), "A Probabilistic Model for Software Defect Prediction", For submission to IEEE Transactions in Software Engineering.

[11] Mark C. Paulk, et. al., "The Capability Maturity Model for Software", Paper, Carnegie Mellon University, pp.1-26.

[12] Paulk, M. C., Weber, C. V., Curtis, B., & Chrissis, M. B. (1993). "Capability maturity model", version 1.1. IEEE Software, 10(3), 18–27.

[13] Schneidewind, N. F. (2001), "Investigation of logistic regression as a discriminant of software quality", In The Proceedings of 7th International Software Metrics Symposium, London, UK, pp. 328–337.

[14] Khoshgoftaar, T. M., & Seliya, N. (2002). Tree-based software quality models for fault prediction. In Proceedings of 8th International Software Metrics Symposium, Ottawa, Ontario, Canada (203–214).

[15] Khoshgoftaar, T. M., & Seliya, N. (2003). Fault prediction modeling for software quality estimation: comparing commonly used techniques. Empirical Software Engineering, 8, 255–283.

[16] Bibi S., Tsoumakas G., Stamelos I., Vlahavas I.(2006), "Software Defect Prediction Using Regression via Classification", IEEE International Conference on Computer Systems and Applications, pp.330 - 336,

[17] Norman Fenton, Paul Krause and Martin Neil, (1999), "A Probabilistic Model for Software Defect Prediction", For submission to IEEE Transactions in Software Engineering.

[18] Ahmet Okutan, Olcay Taner Yıldız,(2012) "Software defect prediction using Bayesian networks", Empir Software

19:154–181 © Springer Science+Business Media, LLC.
[19] Mrinal Singh Rawat, Sanjay Kumar Dubey,(2012) "Software Defect Prediction Models for Quality Improvement: A Literature Study", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, pp 288-296.
[20] Supreet Kaur, and Dinesh Kumar, "Software Fault Prediction in Object Oriented Software Systems Using Density Based Clustering Approach", International Journal of Research in Engineering and Technology (IJRET) Vol. 1 No. 2 March, 2012 ISSN: 2277-4378
[21] Xiao-dong Mu, Rui-hua Chang, Li Zhang, "Software Defect Prediction Based on Competitive Organization CoEvolutionary Algorithm", Journal of Convergence Information Technology(JCIT) Volume7, Number5, 2012.
[22] N. Fenton and M. Neil (2008) "Using Bayesian networks to predict software defects and reliability", Proc. IMechE Vol. 222 Part O: J. Risk and Reliability, pp 702-712.
[23] Jie Xu, [2]Danny Ho and [1]Luiz Fernando Capretz, "An Empirical Study On The Procedure To Derive Software Quality Estimation Models", International journal of computer science & information Technology (IJCSIT) Vol.2, No.4, 2010.
[24] Manu Banga, "Computational Hybrids Towards Software Defect Predictions", International Journal of Scientific Engineering and Technology Volume 2 Issue 5, pp : 311-316, 2013.
[25] Mohamad Mahdi Askari and Vahid Khatibi Bardsiri (2014), " Software Defect Prediction using a High Performance Neural Network", International Journal of Software Engineering and Its Applications Vol. 8, No. 12 (2014), pp. 177-188.
[26] Kamaljit Kaur (2012), "Analysis Of Resilient Back-Propagation For Improving Software Process Control" International Journal of Information Technology and Knowledge Management July-December 2012, Volume 5, No. 2, pp. 377-379.
[27] Mrs.Agasta Adline, Ramachandran. M(2014), "Predicting the Software Fault Using the Method of Genetic Algorithm", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Special Issue 2,, pp 390-398.
[28] Karpagavadivu.K, et.al. (2012), "A Survey of Different Software Fault Prediction Using Data Mining Techniques Methods", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 1, Issue 8, pp 1-3.
[29] Xiaoxing Yang, et.al. (2014), IEEE TRANSACTIONS ON RELIABILITY, This article has been accepted for inclusion in a future issue of this journal.
[30] Ahmet Okutan1 and Olcay Taner Yıldız, (2013), "A Novel Regression Method for Software Defect Prediction with Kernel Methods", ICPRAM 2013 - International Conference on Pattern Recognition Applications and Methods, pp 216-221.
[31] Yajnaseni Dash, Sanjay Kumar Dubey, (2012), " Quality Prediction in Object Oriented System by Using ANN: A Brief Survey", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 2,, pp.1-6.
[32] Ms. Puneet Jai Kaur, Ms. Pallavi, (2013), " Data Mining Techniques for Software Defect Prediction", International Journal of Software and Web Sciences (IJSWS), International Journal of Software and Web Sciences 3(1), pp. 54-57.
[33] Sonali Agarwal and Divya Tomar, (2014), " A Feature Selection Based Model for Software Defect Prediction", International Journal of Advanced Science and Technology Vol.65 (2014), pp.39-58.
[34] Romi Satria Wahono and Nanna Suryana (2013), "Combining Particle Swarm Optimization based Feature Selection and Bagging Technique for Software Defect Prediction", International Journal of Software Engineering and Its Applications Vol.7, No.5 (2013), pp.153-166.
[35] K.Venkata Subba Reddy and Dr.B.Raveendra Babu (2013), "Logistic Regression Approach To Software Reliability Engineering With Failure Prediction", International Journal of Software Engineering & Applications (IJSEA), Vol.4, No.1,, pp. 55-65.