A Systematic Literature Review on Software Fault Prediction based on Qualitative and Quantitative Factors

Gurvinder Singh[1], Baljit Singh Saini[2] & Neeraj Mohan[3]

[1&2]LPU, Phagwara, [3]RBIEBT, Sahauran

Abstract – **The growing demand for higher operational effectiveness and reliability in industrial processes has resulted in a huge attention in fault detection techniques. Researcher and practitioners are remains concerned with correct prediction when developing systems. On the other hand the most popular research area is software fault or fault prediction. Software fault prediction has both security and financial benefits in technical systems by preventing future failures and further improves process upholding schedules. Software fault prediction facilitates to software engineers to attention development activities on defect less code which enhance the software quality and minimize the cost and time to develop software system in today's era of dynamic scenario of globalization. There are many prediction models which are used to filter the software defects. The present study empirically explorers the viability of reducing the software defect prediction based on qualitative and quantitative factors. Further, the study attempts to offer the future prospective in other dimensions like programming languages and for mapping the relation of attributes and fault tolerance.**

Keywords – Software defect, Fault Prediction, Fault Proneness.

I. INTRODUCTION

The ideal prediction of faults in software are expected to occur in code can help direct testing effort, minimize expenses and recover the quality of software. Our aim is to explore how the context models and independent variables used and the modeling techniques functional, manipulate the performance of fault prediction models. A software quality model is a useful tool for meeting the objectives of software reliability and software testing initiatives of different projects [1]. Different Modeling techniques can be used to identify fault free modules [2].

Absence of sufficient tools to guess and evaluate the price for a software system failure is one of the main challenges in software engineering. They used old dataset of software to make and authenticate estimation or prediction system of software development efforts, which allows them to compose management decisions, such as resource allocation. The use of single feature of software to predict faults is not helpful. Fenton uses an example where the same program functionality is achieved using dissimilar programming language constructs resulting in dissimilar static measurements for that module [3]. Fenton uses this example to argue the uselessness of static code attributes. However, where single feature fail and combination succeed [4], Hence combination of static features extracted from requirements and code can be good predictors for identifying modules that actually contains fault.

When we use machine learning methods to make such predication systems, poor data quality in either training set or test set or both sets, can affect prediction accuracy. Various machine learning algorithms has been used in systems engineering to predict faults, software project development effort, software quality and software defects. Evaluation of the use of machine learning in software engineering report that machine learning in software engineering is a mature methods based on mostly available tools using well understood algorithms. The decision tree (DT) classifier is an example of a machine learning algorithm that can be used for predicting continuous attributes (regression) or categorical attributes (classification). Thus, software prediction can be cast as a supervised learning problem, i.e., the process of learning to separate samples from dissimilar classes by finding features between samples of known classes. Software quality models ensure the reliability of the delivered products. It has become vital to develop and implement good software quality models early in the software development life cycle, especially for large scale development efforts.

Software quality prediction models seek to predict quality aspect such as whether a component is fault prone or not. Methods for identifying fault prone software modules support helps to improve resource planning and scheduling as facilitating cost avoidance by efficient verification. Such models can be used to predict the response variable which can either be the class of a module (e.g. fault-prone or not fault-prone) or a quality factor (e.g. number of faults) for a module. The basic hypothesis of software quality prediction is that software currently under development is fault prone if a module with the similar product or process metrics in previous project developed in the same environment was fault free. Therefore, the figures available early within the existing project or from the earlier project can be used in making predictions. This technique is very useful for the large-scale projects or projects with multiple revisions.

II. LITERATURE REVIEW

Statistical methods, machine learning method, and mixed techniques are widely used in literature to predict software faults.

Table-1

Fujimaki (2005)	Koru and Tian (2005)
Bellini et al (2005)	Yan Ma et al. (2007)
Seliya N. et al (2005)	Norman Fenton, et al (2007)
Ceylan E. et al. (2006)	Chaudhary Pree, et al (2012)
Seliya N et al (2007)	Kaur Arashdeep, et al (2011)
Jiang et al.(2007)	Challagulla et al.(2005)

III. SOFTWARE FAULT, FAULT, FAILURE

Faults contain in software systems and it continue to work, is a major problem in future. A software bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from behaving as intended (e.g., producing an incorrect result). A software fault is a deficiency that causes software failure in an executable product. In software engineering, the non-conformance of software to its requirements is commonly called a bug. Most bugs arise from mistakes and errors made by people in either a program's source code or in its design, and a few are caused by compilers producing incorrect object code. Meaningful the causes of possible defects as well as identifying general software process areas that may need attention from the initialization of a project could save capital, time and effort. The possibility of early estimating the potential faultiness of software could help on planning, controlling and executing software development activities. Software is said to be faulty if we feed some input and it produce incorrect output. For each execution of the software program where the output is incorrect, a failure is observed. Software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects but on the other hand fault is a hidden programming error that may or may not actually evident as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software faults are all due to human errors in creating the software. The following depicts the types of failure with its description.

Table-2

Failure class	Description
Transient	Occurs only with certain inputs
Permanent	Occurs with all inputs
Recoverable	System can recover without operator intervention
Unrecoverable	Operator intervention is required to recover from failure
Non-corrupting	Failure does not corrupt system state or data
Corrupting	Failure corrupts system state or data

IV. RATIONALE OF THE STUDY

The increasing demand for higher preparation competency and security in engineering processes has resulted in huge interest in fault-detection techniques. Engineering researchers and practitioners remain concerned with accurate prediction on qualitative and quantitative factors. So it is in this environment Software quality prediction models seek to predict quality factors such as whether a component is fault prone or not, despite earlier attempts to estimate the fault prediction, the less stress has been paid on the reducing of the software defect and predicting it before occurred. Thus the study is a significant attempt that will be helpful for the prediction of faults.

V. OBJECTIVE OF THE STUDY

Find best machine learning algorithm for Software Fault Prediction Based on Quantitative and Qualitative Factors. The Comparison criteria for the different algorithms are based on the Mean absolute error (MAE), Root mean square error (RMSE) Values and accuracy.

VI. RESEARCH METHODOLOGY

The methodology consists of the following steps:

A. *To find the Qualitative and Quantitative attributes of software systems.*

The first step is to find the Qualitative and Quantitative factors of software systems i.e. software metrics. The real-time defect data sets are taken from data repository, available online at: http://promisedata.googlecode.com/svn/trunk/defect/.

B. *To Select the suitable metric values as representation of statement.*

The suitable metrics like product requirement metrics and product module metrics out of these data sets are considered. The term product is used referring to module level data. The term metrics data applies to any finite numeric values, which describe measured qualities and characteristics of a product. The term product refers to anything to which defect data and metrics data can be associated.

C. *Analyze, refine metrics and normalize the metric values and Explore performance of Machine Learning Algorithms.*

In this step the metric values are analyzed, refined and normalized for the better learning. Thereafter, Machine learning algorithm has best result are selected for experimented with that dataset.

VII. COMPARISON OF ALGORITHMS

The comparisons are made on the basis of the more accuracy and least value of MAE and RMSE error values. Accuracy value of the prediction model is the major criteria used for comparison. The mean absolute error is chosen as the standard error. The technique having lower value of mean absolute error is chosen as the best fault prediction technique.

A. *Mean absolute error*

Mean absolute error, MAE is the average of the difference between predicted and actual value in all test cases; it is the average prediction error. The formula for calculating MAE is given in equation

Figure-1

$$\frac{|a_1-c_1|+|a_2-c_2|+\dots+|a_n-c_n|}{n}$$

Assuming that the actual output is a, expected output is c.

B. *Root mean-squared error*

RMSE is frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated. It is just the square root of the mean square error as shown in equation

Figure-2

$$\sqrt{\frac{(a_1-c_1)^2+(a_2-c_2)^2+\dots+(a_n-c_n)^2}{n}}$$

The mean-squared error is one of the most commonly used measures of success for numeric prediction. This value is computed by taking the average of the squared differences between each computed value and its corresponding correct value. The root mean-squared error is simply the square root of the mean-squared-error. The root mean-squared error gives the error value the same dimensionality as the actual and predicted values.

VIII. RESULTS

The first step is to find the structural code and prerequisite attributes of software systems i.e. software metrics. The real-time defect data sets are taken from http://promisedata.org/repository.

A. *Qualitative factors*

The Quantitative factors are grouped under five topics:

1) Specification and Documentation process
2) New Functionality
3) Design and Development process
4) Testing and Rework
5) Project Management

B. *Quantitative Factors*

The following are the Quantitative factors are:

i) Software size: the size, in KLoC of the developed code and the development language.

ii) Effort: development effort measured in person hours for the software development, from specification review to unit test.

The best learning algorithm is implemented in WEKA environment is one such facility which lends a high performance language for technical computing.

The following parameters are used for building the model and the values used in the experiment:

a) CrossVal -- Sets the number of folds for cross validation (1 = leave one out).

b) Debug -- If set to true, classifier may output additional info to the console.

c) Display Rules -- Sets whether rules are to be printed.

d) Evaluation Measure -- The measure used to evaluate the performance of attribute combinations used in the decision table.

e) search -- The search method used to find good attribute combinations for the decision table.

f) useIBk -- Sets whether IBk should be used instead of the majority class.

IX. CONCLUSION

Software Fault prediction is an important topic in software engineering. Software Fault prediction models have the potential to improve the quality of software systems and reduce the expenditure related with delivering those systems. The study evaluated the performance of various machine learning techniques for the fault dataset. Techniques have shown better results than other algorithms with lower values of MAE, RMSE and accuracy is implemented using WEKA.

Despite a set of fault prediction studies, there is need to explore and have more research studies with reliable methodology and practical applicability in other dimension like programming languages and for mapping the relation of features and fault tolerance so that software defects could be forecasted and mitigated at the very genesis

X. REFERENCES

[1] Seliya N., Khoshgoftaar T.M. and Zhong S. (2005), “Analyzing software quality with limited fault-proneness defect data”, in proceedings of the Ninth IEEE international Symposium on High Assurance System Engineering, Germany, pp. 89-98.

[2] Jiang Y., Cukic B. and Menzies T. (2007), “Fault Prediction Using Early Lifecycle Data”. ISSRE 2007, the 18th IEEE Symposium on Software Reliability Engineering, IEEE Computer Society, Sweden, pp. 237-246.

[3] Bezdek J.C., Ehrlich R., and Full W. (1984) “FCM: Fuzzy c-means algorithm”. Computers and Geoscience, Volume: 10, pp. 191-203.

[4] A Systematic Literature Review on FaultPrediction Performance in Software Engineering Tracy Hall, Sarah Beecham, David Bowes, David Gray and Steve Counsell.

[5] Challagula, Bastani B. and Yen (2006). “A Unified framework for Defect Data Analysis using the MBR Technique”. Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), Washington, pp. 39-46.

[6] Ma Y. and Guo L. (2006), “A Statistical Framework for the Prediction of Fault-Proneness”, Product Focused Process Improvement, Edition: First, Publisher: Springer Berlin/Heidelberg, pp. 204-214.

[7] Bellini P. (2005), “Comparing Fault-Proneness Estimation Models”, 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05), China, pp. 205-214.

[8] Challagulla V.U.B., Bastani F.B., Yen I. L. and Paul (2005) “Empirical assessment of machine learning based software defect prediction techniques”, 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, USA, pp. 263-270.

[9] Seliya N., Khoshgoftaar T.M. and Zhong S. (2005), op cit.

[10] Norman Fenton, Martin Neil, William Marsh, Peter Hearty, Lukasz Radlinski, Paul Krause, “Project Data Incorporating Qualitative Factors for Improved Software Defect”, Proceedings of the PROMISE workshop, Year: 2007.

[11] Chaudhary Preeti, Mohan Neeraj, Sandhu Parvinder S. “An Empirical Assessment for Software Defect Forecast on Qualitative and Quantitative Factors using Simple Decision Table Majority Classifier”, International Journal of Research in Engineering and Technology (IJRET) Vol. 1, No. 3, 2012 ISSN 2277 – 4378 189.

❖❖❖