Volume 4, Issue 1, January 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Systematic Literature Survey on Various Techniques for Software Fault Prediction

Aditi Sanyal, Balraj Singh
Department of Computer Science Engineering,
Lovely Professional University, India

Abstract— Software fault prediction is a significant part of software quality assurance. Fault prediction is to expect the possibility that the software contains faults. Fault is a flaw that results in failure. Faults in software systems are major problems that need to be resolved. The fault prediction in software is significant because it can help in directing test effort, reducing cost, and increasing quality of software and its reliability. In this paper we had studied various fault prediction techniques and examined the performance of these techniques.

Keywords: Fault prediction, quality assurance, faults, and reliability.

I. INTRODUCTION

Faults are major problem in software systems that need to be resolved. Fault is a flaw that results in failure. We should have to know the clear difference between bug, fault and failure. Failure is deviation of software actions from the expected outcomes. A fault in software is a flaw that results in failure. Bug occurs when specified requirements of the software do not conform. There are many number of software having number of faults are delivered to the market. So, the main goal is to have software that contains less number of faults as much as possible.

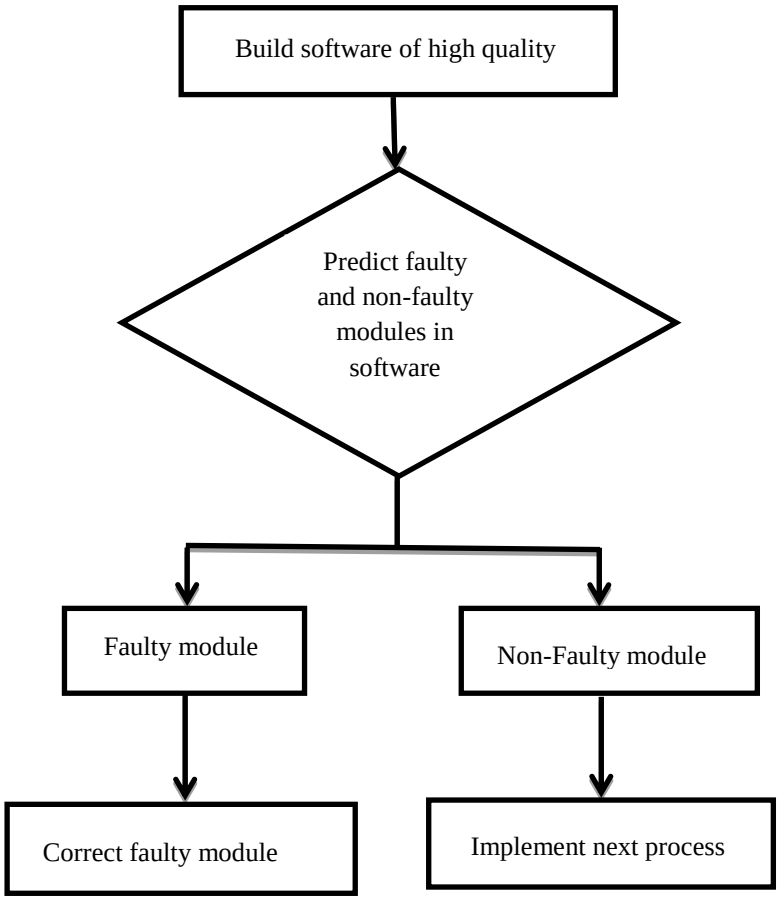

Fig 2.1 Software Fault Prediction

Software defect can be requirement defect, design defect, code defect, test case defect and other product defect.

Software is used in every field of business. So, the software companies' can't danger their business by evolving software of low quality. These threats can be reduced by calculating the quality at the very early stages of development cycle. This would not only decrease the client frustration but also rate of improving faults. If faults are improved after testing, then the rate of correcting defects is high. Fault prediction is to expect the possibility that the software contains faults. To predict whether software contains faulty content or not we use software metric and defect data. To check whether software has better quality and reliability, software faults is to be predicted on time.

If we predicting accurate number of faults in the software, it can help in directing test effort required, reducing cost and increasing cost and also increasing reliability of software. We can say software is of good quality only when there is less number of faults present. If there is less number of faults in the software, it will not only increase the software quality but also its reliability.

A software defect prediction models are developed using previous experiences. These models are developed using software measurement and fault data and then model is validated. For the estimation of the quality of a module that is expressed in number of fault and non-fault modules, these models are employed. Software project teams are employed to assign resources to the modules of poor quality.

II. FAULT PREDICTION TECHNIQUES

1). Decision Tree: Decision trees are great and standard tools for classification and prediction. It produces classifiers in a form of structure of tree where each leaf node represents decision node. In this technique, classification starts from root of the tree and continues to move down until leaf node is reached. Classification helps in classifying faulty and non-faulty modules. Prediction helps in predicting faulty and non-faulty modules. Decision tress helps in developing fault prediction models that predicts faults.

2). Neural Network: Neural Network helps in recognizing patterns from the data set. An artificial neural network is composed of many artificial neurons that are interconnected together according to specific network architecture. The goal of the neural network is to transform the inputs into meaningful outputs. Adaptive Resonance Neural Network is generally used for defect prediction in software systems. It helps in identifying faulty modules very excellently. The benefit of using this technique is that it assists in decreasing effort and cost of developing software.

3). Density based clustering approach: Density Based Clustering is a clustering algorithm. It can be used to estimate the number of faulty and non-fault modules in software system. Clusters are defined as areas of higher density. Following are the parameters used in this:

Eps: supreme radius of the nighbourhood

MinPts: least possible number of points in an Eps-neighbourhood of that point.

4). Bagging method: It creates base learners on many data subsets that are uniformly sampled from the original data, and then uses a linear combination to aggregate them. It is also referred as Bootstrap Aggregating. Combination technique can be majority voting. It also helps in identifying faulty and non-faulty modules with data sets that suffers from imbalance problem. This method can increase the performance of the defect data predictions.

5). Naïve Bayes: It is a classifier based on Bayes theorem used in software fault prediction. It resolves the several difficulties like spam classification (to predict whether email is spam or not), medical diagnosis (given list of symptoms, predict whether patient has cancer or not) and so on. This method can be used to predict faulty and non-faulty modules.

III. LITERATURE SURVEY

(Sunil, Arashdeep, Mehak, & kaur, 2010) had proposed K- Sorensen- means clustering that uses Sorensen distance for calculating cluster distance to predict faults in software. In this paper, data set was collected from NASA MDP repository online. The proposed method is implemented on MATLAB. It took only three projects and information about requirement and code metrics is considered. Author had joined requirement metrics and code metrics to develop a model that called as alliance metrics model. The results showed that the results are more accurate than model developed using K- Canberra- means algorithm and give better fault prediction. Proposed model will thus more correctly categorize components into fault free and fault prone.

(A.A. Shahrjooj Haghighi, 2012) had proposed a fault detection system with higher performance which decreases the cost of software fault detection. He had examined the performance of the 37 different classifiers over 5 public NASA datasets. The proposed technique is implemented on Weka. By comparing different classification algorithms, author figured that Bagging shows higher performance and accuracy compared to others. For verification of the selected classifiers, author had compared the performance of the Bagging, Naïve Bayes and classification via Regression on more datasets. The results demonstrated that Bagging has highest performance on fault detection system. Detection system employed with Bagging is more accurate.

(Catal, 2012) had investigated various performance evaluation metrics and categorized these metrics into two main groups. The first group of metrics evaluates the performance of the prediction system, which classifies the modules into faulty and non-faulty modules and the second group of metrics evaluates the performance of the system which predicts the number of faults in each module of the next release of the system. Metrics from one of these groups can be chosen according to the research objectives. The first group of the metrics is calculated by using confusion matrix and area under ROC curve (AUC). In addition to AUC, PD, and PF, balance metrics are also widely used. Author suggested using the AUC value to evaluate the performance of fault prediction models. From the second group of metrics, R2 and AAE / ARE can be used to ensure the performance of the system that predicts the number for faults.

(Ritika Sharma, 2012) had addressed the problem that large number of faults are present in less number of modules, therefore for improving the software quality, we need to predict number of faults as much as possible. Author had developed and refined techniques that will predict the faults in the software early. In this study, author had used two different metrics that is complexity metrics and object oriented metrics. These metrics help in detecting faulty modules. The result showed that different metrics can be used to achieve higher accuracy.

(seliya, Tagi M. Khoshgoftaar, & Jason Van Hulse, 2010) had addressed the class imbalance problems while developing fault prediction model. He had proposed Roughly Balanced Bagging Algorithm that combines data sampling and bagging into one technique that helps in building software fault prediction model. Classification algorithms i.e. Naïve Bayes and C4.5 decision tree are used by the author. In this paper, software metrics and faulty data sets are considered as data set that is taken from software quality assurance systems. Each data set includes number of faulty, non-fault modules and metrics. The result showed that model based on the Roughly Balanced Bagging is better than the model without bagging or data sampling and overall performance of the Naïve Bayes is better than C4.5 but when combined with Roughly Balanced Bagging, c4.5 performs better than that Naïve Bayes.

(Xu-Ying Liu, 2008) had proposed two algorithms. Deficiency that needs to be overlooked is majority instances are not considered. EasyEnsemble and BalanceCascade were the techniques used by the author to overcome this deficiency. These two algorithms uses the majority class examples that are not by under-sampling. These two algorithms performs better than under-sampling because they better use majority instances. Results showed that these two methods have higher performance than other class-imbalance learning methods.

(Yue Jiang, 2009) had explored five questions regarding fault prediction model: (1). Do different variance occurs when different classification techniques are applied? (2). Does accuracy of the prediction performance depends upon the size of the data set? (3). Does the stability of the model is influenced by the size of the subset collected from the original data set? (4). Do the performance of the different classifiers vary? (5). If you are running cross validation technique 10 and 100 times, do the results vary? Author found that large data set is more informative than small data set. Results confirmed that variance is a very significant feature in accepting fault prediction models.

IV. CONCLUSION

In this paper, we studied various techniques (like decision tree, neural network, naïve Bayes and so on) to predict faults in software systems. The main aim is to examine the performance of different techniques in software fault prediction. Fault prediction using these techniques helps in improving the quality of the software. The fault prediction in software is significant because it can help in directing test effort, reducing cost, and increasing quality of software and its reliability.

RFERENCES

[1] A.A. Shahrjooj Haghighi, M. A., "*Appling Mining Schemes to Software Fault Prediction: A Proposed Approach Aimed at Test Cost Reduction*," Proceedings of the World Congress on Engineering, 2012.

[2] Catal, C., "*Performance Evaluation Metrics for Software Fault Prediction Studies*," Acta Polytechnica Hungarica, 2012.

[3] Ding, Z. (2011). "*Diversified Ensemble Classifiers for Highly Imbalanced Data Learning and their Application in Bioinformatics,"* 2011.

[4] Magdy, A. (n.d.). Retrieved from http://www.slideshare.net/ahmeda1works/a-survey-of-fault-prediction-using-machine-learning-algorithms

[5] Mikel Galar, A. F., "*A Review on Ensembles for the Class Imbalance Problem: Bagging, Boosting and Hybrid Based Approaches*," IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, july. 2012.

[6] Parvinder S. Sandhu, J. S., "*A K-Means Based Clustering Approach for Finding Faulty Modules in Open Source Software Systems*," World Academy of Science, Engineering and Technology 48, 654-658, 2010.

[7] Ritika Sharma, N. B., "*Study of Predicting Fault Prone Software Modules*," International Journal of Advanced Research in Computer Science and Software Engineering, 2012.

[8] S. G., A. K., M. A., & kaur, d., "*A Clustering Algorithm for Software Fault Prediction,"* IEEE international conference on computer and communication technology, (pp. 603-607), 2010.

[9] Sandhu, D. P., & manpreet kaur and amandeep kaur, "*A Density Based Clustering Approach for Early Detection of Fault Prone Modules,"* IEEE International Conference on Electronics and Information Engineering, (pp. 525-530), 2010.

[10] Seema Singh, M. S., "*Software Defect Prediction using Adaptive Neural*," International Journal of Applied Information Systems (IJAIS), 29-33, 2012.

[11] Seliya, N., Tagi M. Khoshgoftaar, & Jason Van Hulse, "*Predicting Faults in High Assurance Software*," IEEE 12th International Symposium on High Assurance Systems Engineering, 2010.

[12] Tracy Halla, S. B, "*A Systematic Review of Fault Prediction Performance*," 2011.

[13] Xu-Ying Liu, J. W.-H., "*Exploratory Undersampling for Class-Imbalance Learning,"* IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS – PART B, 2008.

[14] Yue Jiang, J. L., "*Variance Analysis in Software Fault Prediction Models,"* IEEE 20th International Symposium on Software Reliability Engineering, 2009.

