A Survey on Software Defect Prediction

Bhushan Bharti[#1], Narendra Parmar[*2], Gagan Sharma[#3]

[#]*Department of Computer Science and Engineering, Sri Satya Sai College of Engineering*
Bhopal, India

[1]bhushanbharti76@gmail.com
[2] narendrapcst@gmail.com
[3] deep_325@yahoo.com

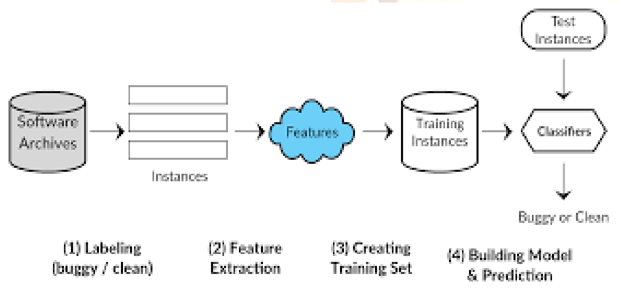
***Abstract*— the demand for software quality has increased tremendously in the last few years giving rise to other concerns like software testing, defect prediction and debugging. This is because software with defects cannot be considered as good software. Having known the possible location of the faults and defects and troubleshooting them in time may save a good amount of time, manpower and money. One of the best and cheapest ways would be to learn from past mistakes to prevent future defects and problems. Several algorithms of Data mining are applied to the various software development tasks by Software Engineers and experts to enhance the software quality & productivity and to decrease the Cost and time of the project.**

Keywords — *Software Defects, Prediction, data mining, classification, regression*

I. Introduction

Although there is variety in the definition of software quality, it is truly accepted that a project with many defects lacks the quality of the software. Knowing 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 money, time and working effort.

There have been several data mining methods for analysis of defects over the last few years, but very few of those techniques are capable of properly dealing with the above mentioned concerns. Many analysts and experts have been using different methods with multiple combinations of various data sets to predict possible faults. Software with defects is largely considered as poor quality software. Software Experts and developers use various algorithms of data mining to figure out potential errors and bugs that can lead to software failure in future.



Common Process of Software Defect Prediction

Software defect Prediction plays a very crucial role in software development life cycle. A defect predictor is a method or tool which helps in prediction of possible defects in possible locations beforehand. As per the experts a big part of the software development cost lies in testing. Testing phase is as much as half of the complete life cycle of the whole project. Therefore, testing phase becomes our main challenge to find out bugs, errors and defects and identify their proper location before the actual testing of the software begins. This helps the developers to effectively share their their resources and resource person to those particular tasks.

II. Literature survey

Our main aim shall be to discuss the work done in the area of software defect prediction and finding out possible solutions to avoid or correct those defects and faults. This is important because software with defects is largely considered as poor quality software.

In 2006, Bibi, Tsoumakas, Stamelos, Vlahavas, applied an approach of machine learning to the defects estimation problem which they called as the Regression via Classification (RvC) [1].The whole process of Regression via Classification (RvC) comprises two important stages

1. It is a method in which the problem of classification is turned into a problem of classification. The target values are converted into classes using a process of discretization.
2. The class output is then reversed to find out the prediction in numerical form.

Menzies, Greenwald, and Frank (MGF) [2] suggesting in their research in 2007 where Rule Induction and Naïve Bayes machine learning algorithms were compared and their performances were analyzed to pre determine the possible defects in components of software under study.

In 2007, MLT(Multilayer Perceptron), Voting feature Intervals(VFI) and NB were used by Oral and Bener [3] for prediction of Embedded Software Defects using seven sets of data for their research

In 2007, Iker Gondra [4] used a machine learning methods for defect prediction. He used Artificial neural network as a machine learner.

.

In 2011, CBA i.e. Classification based Association was used or prediction of Defects in software by Baojun, Karel [5]. The rules generated for CBA-RG classification Association Rules

III. SOFTWARE DEFECT PREDICTION CHALLENGES

Developing zero bug software for commercial purpose is always a big challenge. Since licences and commercial software do not share their data due to their policies it becomes difficult to access all types of data set. Building quality software defect prediction model with heterogeneous dataset and cross project is quite difficult task [6]. It is always a big challenge for developers to predict the defects with software that require continuous updating as changes are made to the codes quite often. Data sets are also often changing which leads to difficulty in defect prediction. However, proprietary datasets are not publicly available because of privacy issue [7].

IV. DEFECT PREDICTION METRICS

Defect prediction metrics play the most important role to build a statistical prediction model. Most defect prediction metrics can be categorized into two kinds: code metrics and process metrics. Code metrics are directly collected existing source code while process metrics are collected from historical information archived in various software repositories such as version control and issue tracking systems[8].

a. Code Metrics
Code metrics also known as product metrics measure complexity of source code. Its ground assumption is that complexity source is more bug-prone. To measure code complexity, researchers proposed various metrics.

b. Process Metrics
Below is the list of seven representative process metrics that are often used in developing prediction model. [9][10][11][12][13]
1. Relative Code Change Churn
2. Change Metrics
3. Change Entropy
4. Code Metric Churn/Code Entropy
5. Popularity
6. Ownership and authorship
7. Micro Interaction Metrics.

There are lots of debates if code metrics are good defect predictors and process metrics are better than code metrics. Menzies et al. confirmed that code metrics are still useful to build a defect prediction model [2]. However, according to Rahman et al.'s recent study comparing code and process metrics, code metrics is less useful than process metrics because of stagnation of code metrics [14].

V. GENERAL PROCESS OF SOFTWARE DEFECT PREDICTION

To build an efficient prediction model, we should have proper data on defects and metrics, which can be accumulated from software development efforts to use as the learning set. Thus, there is trade-off between its prediction performance on additional data sets and how well this model fits in its learning set. Therefore, the performance of the model is assessed by the comparison of the predicted defects of the modules in a test, against the actual defects witnessed [15]

The steps involved in Software Defect Prediction Process are as follows.
1. Labelling.
2. Extracting features and creating training data sets.
3. Develop a prediction model.
4. Final Assessment.

VI. APPLICATIONS OF DEFECT PREDICTION

One of significant objectives of defect prediction models is efficient utilization of available resources for assessing and testing programming modules. Nevertheless, there is only a hand few of contextual analyses which use defect prediction models [16]. Thus, Rahman et al. [17] led most of their investigation on cost-viability. Lewis pioneered a recent contextual investigation directed by Google, which compares the BugCache and Rahman's technique, with respect to the amount of closed bugs[18]. The outcomes have indicated that the designers favored Rahman's technique. In any case, the defect prediction models do not give any advantages to the developers. In a recent survey, Rahman et al demonstrated that defect prediction models could be useful to organize potential warnings discovered by the bug finders, for example, FindBug. It also helps in implementation of results from the defect prediction to organize or choose appropriate test cases. In regression testing, performing all the test cases are not financially feasible, and consumes large amount of time as well. Therefore, it is best to choose proper test cases, which investigates the potential faults in the system [19]. The results of the defect prediction models can provide an idea on the potential bugs and their severity, which can be exploited to select and prioritize the test cases. On the basis of previously reviewed works, it is obviously that the area of defect prediction has more to offer, and hence, it is in its early stages. It can be concluded with few of the future improvements and limitation, which can be extracted from past research works.

Software defect prediction aims to reduce software testing efforts by guiding testers through the defect-prone sections of software systems. Defect predictors are widely used in organizations to predict defects in order to save time and effort as an alternative to other techniques such as manual code reviews.

VII. CONCLUSION

This survey paper helps the researchers to study about software defects and software defect prediction techniques. To implement the data pre-processing technique; data cleaning, data normalization and data discretization will be performed in data mining. For feature extraction and selection to implement of new approach, to implement of evolutionary computation and optimization technique for best feature selection and to implement machine learning classification techniques for bug classification. An improved approach consists of data pre-processing low computation cost, complex model, software defect prediction comparative analysis and improved classification performance of the system

VIII. REFERENCES.

1. S Bibi, G Tsoumakas, I Stamelos, and I Vlahavas. Software defect prediction using regression via classification. In IEEE International Conference on, pages 330–336, 2006.
2. Tim Menzies, Jeremy Greenwald, and Art Frank. Data mining static code attributes to learn defect predictors. Software Engineering, IEEE Transactions on, 33(1):2–13, 2007
3. Ata¸c Deniz Oral and Ay¸se Ba¸sar Bener. Defect prediction for embedded software. In Computer and information sciences, 2007. iscis 2007. 22nd international symposium on, pages 1–6. IEEE, 2007.
4. Iker Gondra. Applying machine learning to software fault-proneness prediction. Journal of Systems and Software, 81(2):186–195, 2008.
5. Ma Baojun, Karel Dejaeger, Jan Vanthienen, and Bart Baesens. Software defect prediction based on association rule classification. Available at SSRN 1785381, 2011.
6. S. J. Pan and Q. Yang. A survey on transfer learning. IEEE Trans. on Knowl. and Data Eng., 22:1345–1359, October 2010
7. F. Peters and T. Menzies. Privacy and utility for defect prediction: Experiments with morph. In Proceedings of the 34th International Conference on Software Engineering, ICSE ’12, pages 189–199, Piscataway, NJ, USA, 2012. IEEE Press
8. S. D. Conte, H. E. Dunsmore, and V. Y. Shen. Software Engineering Metrics and Models. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1986.
9. N. Nagappan and T. Ball. Use of relative code churn measures to predict system defect density. In Proceedings of the 27th international conference on Software engineering, ICSE ’05, pages 284–292, 2005.
10. R. Moser, W. Pedrycz, and G. Succi. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In Proceedings of the 30th international conference on Software engineering, ICSE ’08, pages 181–190, 2008
11. A. E. Hassan. Predicting faults using the complexity of code changes. In Proceedings of the 31st International Conference on Software Engineering, ICSE ’09, pages 78–88, 2009.
12. M. D’Ambros, M. Lanza, and R. Robbes. An extensive comparison of bug prediction approaches. In Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on, pages 31 –41, May 2010.
13. A. Bacchelli, M. D’Ambros, and M. Lanza. Are popular classes more defect prone? In Proceedings of the 13th International Conference on Fundamental Approaches to Software Engineering, FASE’10, pages 59–73, Berlin, Heidelberg, 2010. Springer-Verlag
14. F. Rahman and P. Devanbu. How, and why, process metrics are better. In Proceedings of the 2013 International Conference on Software Engineering, ICSE ’13, pages 432–441, Piscataway, NJ, USA, 2013. IEEE Press.
15. Hewett, R. 2011. Mining software defect data to support software testing management. Applied Intelligence, 34, 245–257.
16. Lewis, N.D. 1999. Assessing the evidence from the use of SPC in monitoring, predicting & improving software quality. Computers & Industrial Engineering, 37, 157–160.
17. Rahman, F., Posnett, D., Devanbu, P. 2012, November. Recalling the imprecision of cross-project defect prediction. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, 61. ACM.
18. Peters, F., Menzies, T., Marcus, A. 2013, May. Better cross company defect prediction. In Proceedings of the 10th Working Conference on Mining Software Repositories, 409–418. IEEE Press
19. Lessmann, S., Baesens, B., Mues, C., Pietsch, S. 2008. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Transactions on Software Engineering, 34, 485–496.