Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 9, September 2015, pg.250 – 254

REVIEW ARTICLE

SOFTWARE FAULT PREDICTION: A REVIEW

RAKESH KUMAR
M.Tech. Scholar, Dept. of CSE
KNIT, Sultanpur U.P.
rkyknit@gmail.com

D.L. Gupta
Associate Prof., Dept. of CSE
KNIT Sultanpur U.P.
dlgupta2002@gmail.com

ABSTRACT- Software defect prediction in software engineering is one of the most interesting research fields. To improve the quality and reliability of the software in less time and in minimum cost, it is the most relevant key area where various researchers have been done. When the size and complexity of software increases then faults prediction in the software became more difficult. To maintain the high level of quality of the software, there is need of a model or system which can classify the software in two prone modules as faulty and non-faulty prone. In the process of predicting faulty and non-faulty prone, the prediction of faulty prone modules incurs more cost and time than prediction of non-faulty prone modules.

In this literature survey, the process of defect prediction is analyzed. There are different techniques to predict faults and evaluate performance of the predictors. These predictors may be a model, system, techniques or algorithm. In this review it is studied that what type of progress has been done up to now and which type of software metrics have been used to design the fault predictor. It is also discuss about the cross project fault prediction which are more demanding in today's scenario.

Keywords– Software Bug prediction, Public Dataset, Neural Network

1. INTRODUCTION

Software defect prediction is a key process in software engineering to improve the quality and assurance of software in less time and minimum cost. It is implemented before the testing phase of the software development life cycle. Software defect prediction models provide defects or no. of defects. Software defect prediction has been motivated to a number of researchers to provide different model with a project or cross project to improve various quality and monitoring assurance of software. There are two approaches to build a software defect prediction model like supervised learning and unsupervised learning. Supervised learning has the problem that to train the software defect prediction model need the historical data or some known results. The training of the model within the project is performed well but it causes challenging problem in understanding of other new projects. There are many public datasets which are available free for the researcher like PROMISE, Eclips and Apache to overcome the challenging problem when training performed on new project. The different researcher have been taken interest to build a cross project defect prediction model with a number of metrics set like class level metric, process metrics, static code metrics but they could not build more feasible accurate models. There are many classifiers or learning algorithm to select a wide variety of software metrics like Naïve Bias, Support Vector Machine, Random Tree, J48 and Logistic Regression. These classifiers have achieved many useful conclusions. Almost all the present software prediction models have been built using complex metrics by which the prediction model achieved the satisfactory accuracy. In this paper the contribution is relate to the current state of research. It also suggested the prediction model with the simplified set of metrics for feature selection. It also demonstrated that the software prediction model build with minimum set metrics can achieve the acceptable result.

This survey is related to many number of research paper published in recent 10 years in different publication like IEEE transaction, international journal, international conferences. This paper is organized as the following section. As section one is related with introduction. Section 2 is providing the information about defect prediction. Section 3 provides information about the evolution of software defect prediction models. In section 4 defect prediction metrics are described. Section 5 provides information about defect prediction models. Section 6 gives the details about the preprocessing techniques. Cross project defect prediction approach provide in section 7. The next section 8 describes about different application of defect prediction model. Section 9 provides challenges and future scope.

2. SOFTWARE DEFECT PREDICTION PROCESS

The common software defect prediction process follow the machine learning approach, [1][5][16]. The first step in predict process to find out the instances from software, an instance can be code, function, class or method etc. These instances can be generate from the different issue tracking system, version control system or e-mail archives. An instance has different metrics which is find out from the software. These instances can be categorized in buggy B or number of bugs and clean C or number of clean.

After recognizing instances with the category and metrics, the first step of machine learning preprocessing techniques used on instances to create new same type of instance. The preprocessing is applied to extract the features, scaling the data and removing the noise [18][25][10]. It is not compulsory to apply on all type of defect prediction models [5][22]. After preprocessing the instances generated new instances to train the defect prediction model. The prediction model provides the result in terms of buggy instances and clean instances. The number of bugs in an instance is known as regression. It produces only two results for the instances buggy or clean so it is also known as binary classification.

3. EVOLUTION OF SOFTWARE DEFECT PREDICTION

In 1971, Akiyama [3] firstly conduct the analysis on approximating the number of defects and realize that complex source code caused more number of defects. He thought that the large software has many line of code so LOC can measure the complexity of software. So he assumed to the LOC as the metric. He has designed the firs defect prediction model based on the LOC Metrics. But it is analyzed that LOC metric is too small metric to measure the complexity of any software. So to overcome this problem, Halested and McCabe in 1977 and 1976 proposed the Halsted complexity metric and cyclomatic complexity metric respectively [13][24].
It is analyzed in the period of 1970s to 1980, it was become popular for estimate the number of defects but it was not really a prediction model. It was a simple fitting model which provides the correlation between defects and metrics [15]. This fitting model failed to validate new module of software.

So to short out this limitation of defect prediction model an active researcher Shen et al. created a model based on linear regression and also test the model for new module of software [28]. But Munson et al had been stated that the regression techniques is not the precise and proposed a new defect prediction model based on classification techniques which classify the model in to two parts low risk and high risk[43] and achieved the accuracy of 92%. This model has the limitation that it had not any metrics of object oriented system and having few resources for further development.

In considering the object oriented system. Kemerer and chidamber in 1994 [21] were proposed many object oriented metric and in 1990 [4] Basili et al. proposed a new defect prediction model based on object oriented metrics. In the recent year of 2000, the various process metrics were evaluated [1][3][11].

But in year 2000, there were various limitations for defect prediction model. It was not validate the prediction after product release to assure the software quality. The defect prediction model could not capable to predict defects whenever source code modification performs. So to overcome this problem , Mockus et al. proposed model for changes [2]. This was known as just-in-time (JIT) defect prediction model. JIT model had been studied further by different researches to improve the prediction for change occurred.

The other limitation of software defect prediction model was to create a defect prediction model for new arrival project or software or software with few historical data. To overcome this problem researcher had done various studies to build cross project defect prediction model [29][10]. It raised issue of cross defect prediction identification. The process metric was popular to resolve it but not fully succeeds. So Zimmermann et al. studied the issue and try to make cross project defect prediction model with identifying cross prediction and try to make it more feasible[27][30].

The other limitation of software defect prediction model was that the defect prediction model would be valid when it would be used by the industry. So there were many researchers work on this problem, they studied about case study and different application used to validate it practically to resolve this problem [5][20][8].

Pinzer et al. [14], Taba et al.[17] and Zimmerman et al [26] had studied and analyzed about modern trend of information technology by social network analysis and by network measures and they proposed new concept of prediction model personalized defect prediction model [22] and another was the universal software defect prediction model[9].

4. DEFECT PREDICTION METRICS

The number of researcher has been studied many metrics and proposed different model based on different metrics. Each researcher was proposing new metrics to create the defect prediction model. The mostly used metrics are line of code, source code and process metrics.

Source code- provides the information about the complexity of the software and stated that if source code is big then it would be complex and cause a no. of defects. The process metrics- stated the information about the development process, like interrelation or correlation, right of source code and modification in source code.

Code metrics are directly related to the source code available where process metric is related with historical information archived. Code metric is also told as product metrics which is used to measure the complexity of source code. The different metric used are size metric which measure length, volume, quantity of software product.

It is already told in previous section that most studied base on the machine learning approach or statically approach. The machine based model provides the information about the defect prone in source code known as classification or number of defects in source code known as regression.

Kim et al. proposed a model based on bug cach algorithm. It is different than machine learning approaches. The main working theme of bug cache algorithm, it stored the list of locality information for previous most bug prone source code, methods or files [19].

The researcher also studied the preprocessing techniques used before the creating the model. The preprocessing technique is the important part of defect prediction model. To improve the assurance and quality, the preprocessing techniques used for feature extraction, normalization and noise minimization [18][63].

The other most important studied perform by the researcher was cross project defect prediction which was not feasible for the new arrival software module, only few model had been achieved very less feasibility. But it was too low to accept. Various researchers further studied about the feasibility of cross project defect prediction and stated that to achieve feasibility is hard [30].

5. APPLICATION OF DEFECT PREDICTION

There are many application of software defect prediction. Its main goal is to allocate resources effectively for testing the software products. The case study based software defect prediction model very less used in the industry [5] [4]. Lewis et al. [4] conducted a case study in Google. Rehman et al. also conducted many case study but by these study developer did not get acceptable defect prediction model [4].

Defect prediction could be benefits to prioritize warning by find bug. These study conducted by Rehman et al. [8].

Another application is to prioritize or extract test case. Regression test is costly for all test suits than many prioritizations and selection for test case. Defect prediction model produce the defect prone software and its ranks.

6. THE OTHER EMERGING TOPICS

Apart from the previous section discussion there are other emerging and interesting topics in defect prediction to be study and analyzing. The first one is defect data privacy [7] and the second one is the study about comparison between static defect prediction models.

7. CHALLENGING ISSUES

Defect prediction studies need more implementation and analysis to overcome the challenging issues. It is difficult to apply these approaches practically due to following reasons.

Most of the studied is practically implemented using open data source or public data set so it may not work better for commercial or private dataset. Due to privacy issues the proprietary data are not publically available.

The MORPH algorithm introduce by the Peters et al. to increase the privacy of data which was not validate for the cross project defect prediction [7]. Analyzing these, it can be concluded that if the proprietary data is more available then proposed prediction model then it will be more accurate for cross project defect prediction. Due to different feature space and feasibility study, the cross defect prediction is not easy.

Different feature space - There are many open dataset or public data set available but each data set have not the similar type metric or same no. of metrics. The metrics are evaluated from different domains. So defect prediction model created based on object oriented metric is not applicable for different metric or feature space.

Feasibility - The feasibility of cross prediction model is not more acceptable to make more feasible. Cross project prediction model can become more powerful for the industry.

Defect prediction models which are proposed up to now were not guarantee for good prediction result or performance. As the software repository evolve more new type of development process which never used for software defect prediction models or metrics.

There is need of more study on new metric and modern evolution to make more performable and acceptable defect prediction models.

8. REFERENCES

1. 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.
2. A. Mockus and L. G. Votta. Identifying reasons for software changes using historic databases. In Proceedings of the International Conference on Software Maintenance, 2000.
3. C. Bird, N. Nagappan, B. Murphy, H. Gall, and P. Devanbu. Don't touch my code!: Examining the effects of ownership on software quality. In Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ESEC/FSE '11, pages 4–14, New York, NY, USA, 2011. ACM.
4. C. Lewis, Z. Lin, C. Sadowski, X. Zhu, R. Ou, and E. J. W. Jr. Does bug prediction support human developers? Findings from a google case study. In International Conference on Software Engineering (ICSE), 2013.
5. E. Engstrom, P. Runeson, and G. Wikstrand. An empirical evaluation of regression testing based on fix-cache ¨ recommendations. In Software Testing, Verification and Validation (ICST), 2010 Third International Conference on, pages 75–78, April 2010.
6. F. Akiyama. An Example of Software System Debugging. In Proceedings of the International Federation of Information Processing Societies Congress, pages 353–359, 1971.
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. F. Rahman and P. Devanbu. Comparing static bug finders and statistical prediction. In Proceedings of the 2014 International Conference on Software Engineering, ICSE '14, 2014.
9. F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou. Towards building a universal defect prediction model. In Proceedings of the 11th Working Conference on Mining Software Repositories, MSR 2014, pages 182–191, New York, NY, USA, 2014. ACM. 34
10. J. Nam, S. J. Pan, and S. Kim. Transfer defect learning. In Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, pages 382–391, Piscataway, NJ, USA, 2013. IEEE Press.
11. 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.
12. M. D'Ambros, M. Lanza, and R. Robbes. Evaluating defect prediction approaches: A benchmark and an extensive comparison. Empirical Softw. Engg., 17(4-5):531–577, Aug. 2012.
13. M. H. Halstead. Elements of Software Science (Operating and Programming Systems Series). Elsevier Science Inc., New York, NY, USA, 1977.
14. M. Pinzger, N. Nagappan, and B. Murphy. Can developer-module networks predict failures? In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, SIGSOFT '08/FSE-16, pages 2–12, New York, NY, USA, 2008. ACM.
15. N. Fenton and M. Neil. A critique of software defect prediction models. Software Engineering, IEEE Transactions on, 25(5):675 –689, sep/oct 1999.

16. 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.
17. S. E. S. Taba, F. Khomh, Y. Zou, A. E. Hassan, and M. Nagappan. Predicting bugs using antipatterns. In ICSM, pages 270–279, 2013.
18. S. Kim, H. Zhang, R. Wu, and L. Gong. Dealing with noise in defect prediction. In Proceeding of the 33rd international conference on Software engineering, ICSE ’11, pages 481–490, New York, NY, USA, 2011. ACM.
19. S. Kim, T. Zimmermann, E. J. Whitehead Jr., and A. Zeller. Predicting faults from cached history. In Proceedings of the 29th international conference on Software Engineering, ICSE ’07, pages 489–498, 2007.
20. S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. Software Engineering, IEEE Transactions on, 34(4):485–496, July 2008.
21. S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. IEEE Trans. Softw. Eng., 20:476–493, June 1994.
22. T. Jiang, L. Tan, and S. Kim. Personalized defect prediction. In Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on, pages 279–289, Nov 2013.
23. T. Lee, J. Nam, D. Han, S. Kim, and I. P. Hoh. Micro interaction metrics for defect prediction. In SIGSOFT ’11/FSE-19: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, 2011.
24. T. McCabe. A complexity measure. Software Engineering, IEEE Transactions on, SE-2(4):308–320, Dec 1976.
25. T. Menzies, J. Greenwald, and A. Frank. Data mining static code attributes to learn defect predictors. IEEE Trans. Softw. Eng., 33:2–13, January 2007.
26. T. Zimmermann and N. Nagappan. Predicting defects using network analysis on dependency graphs. In Proceedings of the 30th international conference on Software engineering, ICSE ’08, pages 531–540, 2008.
27. T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ESEC/FSE ’09, pages 91–100, New York, NY, USA, 2009. ACM.
28. V. Y. Shen, T.-J. Yu, S. M. Thebaut, and L. R. Paulsen. Identifying error-prone software an empirical study. IEEE Trans. Softw. Eng., 11(4):317–324, Apr. 1985.
29. Y. Ma, G. Luo, X. Zeng, and A. Chen. Transfer learning for cross-company software defect prediction. Inf. Softw. Technol., 54(3):248–256, Mar. 2012.
30. Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang. An investigation on the feasibility of cross-project defect prediction. Automated Software Engineering, 19(2):167–199, 2012.