Software Defect Forecasting Based on Classification Rule Mining: A Survey

Nafeesa Hamid[1], Jyoti Arora[2],
[1]M.Tech Student, Desh Bhagat University,Mandi Gobindgarh
[2] Assistant Professor, Desh Bhagat University,Mandi Gobindgarh
(E-mail: nafisa_hamid@yahoo.com)

Abstract—There has been rapid growth of software development. Due to various causes, the software comes with many defects. In Software development process, testing of software is the main phase which reduces the defects of the software. If a developer or a tester can predict the software defects properly then, it reduces the cost, time and effort. In this paper, we show a comparative analysis of software defect prediction based on classification rule mining.

Keywords—*Software defect prediction; classification Algorithm; Rule-based Prediction, Software system;*

I. Introduction

There has been a huge growth in the demand for software quality during recent ages. As a consequence, issues are related to testing, becoming increasingly critical. The ability to measure software defect can be extremely important for minimizing cost and improving the overall effectiveness of the testing process. The major amount of faults in a software system is found in a few of its components. 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. The possibility of early estimating the probable faultiness of software could help on planning, controlling and executing software development activities. A low cost method for defect analysis is learning from past mistakes to prevent future ones. Today, there exist several data sets that could be mined in order to discover useful knowledge regarding defects. Using this knowledge one should ideally be able to: a. Identify potential fault-prone software. b. Estimate the distinct number of faults, c. Discover the possible causes of faults.

Defects are basic properties of a system. They come from design or manufacture, or external environment. The systems which run well at the moment may also have defects not trigged now or not so important at the moment. Software defects are programming errors which cause the different behavior compared with expectation. Most of the defects are from source code or deign, some of them are from the wrong code generating from compilers. For software developers and users, software defects are a headache problem. Software defects not only reduce software quality, increase costing but also suspend the development schedule. No matter in software engineering or in research area, to control the number of defects is an important aspect. Finding and fixing the bugs cost lots of money. The data of US department of defense shows that in 2006, American spent around 780 billion dollars for software bugs related problem. And it also shows that there is around 42% money spend on software bugs in IT products [1]. Until now there is no similar report in China, but there is an estimation that the cost for software bugs account for 30% of the whole cost. So it is very worktable to research the software defects.

The defect prevention method does not always prevent defects in the application below test because the application is so complex and impracticable to identify all the errors or faults. The defect detection technology complements the defect prevention effort and uses both methods together to enhance the likelihood that the test team will achieve the identified test objectives and goals. The presence of "defect prevention strategies" not simply reflects anelevated level of test field maturity, but also represents the most cost-effective expenditure associated with overall testing efforts. A variety of methods, tools, techniques and methods to prevent defects are proposed, but they all seem to be insufficient n accurate prediction. More work is still to be adopted to prevent defects in terms of technology and the schemes that are used. In the case of errors detected in the development lifecycle, requirements specifications it can be prevented errors from migrating from design and design to code. Defect prevention is critical to the quality of the organization. The main purpose of quality costs is not to decrease costs but to provide costs in appropriate investments. It should not be delighted as a waste of time while stipulating deep participation. Instead, it should consider saving time, money, and resources it needs. It can save as many reworks as it needs when defects appear in the final or post-delivery period. At every stage of the software lifecycle, defect prevention should be introduced to prevent failures early, take corrective action to eliminate them and avoid their recurrence. A software defect prediction framework is a system that can predict whether a given software module is defective. Typically, software failure prediction models are trained utilizing software measures and fault data composed from earlier developed software releases or related projects. Models can be applied to program modules with unknown defect data.

Usually during the development process, software development team can only know about the software defects by software testing results. But it is expensive to testing the whole process completely, at the same time most of software testing happens at the later stage of software development. If during testing process we find the number Software defects predicting is proposed to solve this kind of problem. The assumption is that the quantity of software is related with the complexity of software modules. More complex modules normally contain more bugs. We can use the historical data, the already discovered software defects and other metric data which can represent software product, software development process to predict the software defects quantity and decide whether the module is defects-prone. In this case, software development team can allocate resources to high risk software module to improve the reliability and quality. By adopting software defects prediction, software development team can forecast the costing in early stage of software development at a relatively lower cost. This will help software development team to optimize allocation of project resources, also help to improve the quality of software. Most of software development teams have these four kind of data, including source code, requirements documentations, testing documentations, defects tracking system. All of the data can be called software development repository. As data mining technique becomes mature and important, also the significant influence it has to the information discovery. Researchers adopt data mining techniques into software development repository to gain the better understanding of software development process, the evolution of software development, to analyze software defects and reuse software modules.

When there repeatedly exists a software failure in system through time it automatically leads to software defect. Software defect are an error that are introduced by software developer and stakeholders. The main objective of software defect prediction is to improve the quality, minimized cost and time of software products. Software defect is also referred to as bug can be defined as shortage in the software product that causes the software not to perform its task as the programmer and customer needed.

II. RELATED WORK

In 2006, Bibi, Tsoumakas, Stamelos, Vlahavas, apply a machine learning approach to the problem of estimating the number of defects called Regression via Classification (RvC) [1].The whole process of Regression via Classification (RvC) comprises two important stages: a) The discretization of the numeric target variable in order to learn a classification model, b) the reverse process of transforming the class output of the model into a numeric prediction.

Menzies, Greenwald, and Frank (MGF) [2] published a study in this journal in 2007 in which they compared the performance of two machine learning techniques (Rule Induction and Naive Bayes) to predict software components containing defects. To do this, they used the NASA MDP repository, which, at the time of their research, contained 10 separate data sets.

In 2007, Iker Gondra [3]used a machine learning methods for defect prediction. He used Artificial neural network as a machine learner. Embedded software defect prediction In 2007, Oral and Bener [4] used Multilayer Perception (MLP), NB, VFI(Voting Feature Intervals) for Embedded software defect prediction. There they used only 7 data sets for evaluation.

In 2011 Baojun, Karel [3] used classification based association rule named CBA2 for software defect prediction. In these research they used association rule for classification. and they compare with other classification rules such as C4.5 and Ripper.

In 2011, Song, Jia, Ying, and Liu propased a general framework for software defect-pronness prediction. in this research they use M*N cross validation with the dataset(NASA, Softlab Dataset) for learining process. and they used 3 classification algorithms(Naive baysed, OneR, J48). and they copared with MGF [2] framework.

Software defect prevention proposals are mainly based on tools, techniques, methods and standards [12], [18]. This is one of the most active areas of research in software engineering, [10], [20], [11], [18], [19], [16]. Because the defect prediction model provides a list of buggy software artifacts, QA teams can efficiently assign limited resources to test and investigate software products [11], [20], [16].

Defect analysis at the early stage reduces time [7], cost, cost, and resources essential. Knowledge of entering faults and process can prevent defects. The study of this knowledge will improve quality and analyze the root cause of defects can prevent the occurrence of defects. Analysis of the main reasons may take two types: "logical analysis" and "statistical analysis". Logical analysis is a humanorientedinvestigation that needs specialized knowledge in products, processes, improvement and the environment. Checks logical connection among errors (effects) and error (reason), and statistical analysis based on empirical learning of similar projects or projects generally written . There are many ways to detect defects such as "inspection", "prototype", "testing" and "accuracy calibration" [8]. Formal testing is the most effective and expensive method of quality control to detect defects at an early stage of development [9], [10]. Prototyping understands the specific requirements to help eliminate some of the shortcomings in defect elimination. Testing is one of the most effectual techniques. It can escape through the early detection of defects [11] which can be detected during the test. Improve accuracy, especially in the coding phase, to determine the best way to go. Precision tuning is the most effective and economical way to create software. Defect prevention can be accomplished by automating the development process. Several tools are offered to analyze the necessity of the stage. The tools available are the requirements for being too costly. It can automate the compliance checks, but this cannot be an automatic integrity check. The tools used in this step include requirements management tools, recorder requirements tools, requirements and validation tools. Design tools include "database design tools", "application design tools", and "visual modelling tools" such as "Rational Rose". Even tools such as "code generation tools", "code testing tools", and "code coverage analysis tools" can be used to automate testing steps. Several tools such as "defect tracking tools", "configuration management tools", and "test procedure generation tools" are available at every stage of development.

Many defect prediction models are based on "machine learning". Depending on what to predict the machine learning models fall into two forms: "classification" and "regression". As the innovative machine learning techniques are being developed, "active or semi-supervised learning methods" that were used to build a good defect prediction models [15], [16]. In addition to machine learning models or statistical models, such as "BugCache" [19] have been projected. The Figure.1 illustrates the frequency of use of the "defect prediction model" in representative defect prediction in the literature [5]. Because "statistical models" based on machine learning have been considered for anextensive time, "classification and regression models" dominate. In the proposed BugCache , there have been several studies examining the BugCache model as well as case studies in [13],[14],[17].

Kim et al. [23] suggested a new "defect prediction model" termed as "change classification". Change classification can be directly beneficial to developers, as opposed to the common failure prediction model because the change classification model can provide immediate predictions every time a developer changes to source code files and commits to the "version control system" [20]. Though, the modified classification model is besideintense for actual use because the model consists of more than 10,000 features [17]. Turhan et al. [22] implemented a nearest neighbour filter applied (NN filter) is used to improve intercompany fault prediction performance. The basic idea behind NN filters is to accumulate related source instances in the objective instance to learn the prediction model. In other terms, if it can build a prediction model utilize a selected source instance with data characteristics similar to the target instance, the model can be better performed when predicting the target instance over the model learned to utilize all source instances. The NN filter selects 10 source illustrations for each target instance as the nearest neighbours. To evaluate the performance of inter-company fault prediction utilizing NN filters were conducted utilizing 10 proprietary data sets from NASA and SOFTLAB [22].

Most existing work on troubleshooting depends on declarative specification rules [6] [7] [8] [5]. These conditions usually determined manually identify the main features that characterize a defect, especially utilizing a combination of quantitative (metric), structural and/or lexical information. However, in a deep scenario, the number of possible defects that can be described manually with the rules can be very large. Dimensions software typically utilized to analyze method efficiency and product software quality for the projects. Failure assessment is carried metrics software and effectively used to predict faults. For each fault, the rule represented by the metric combination requires significant remediation to find the threshold appropriate for each metric. The software is a complex object that consists of different modules with varying degrees of defect frequency. Therefore, it is significant to predict a defective software module before it deploys a software project to plan anim proved up holding schemes. Premature knowledge of faulty software modules can facilitate it plan efficient process improvement at a reasonable time and cost. This can direct to enhanced software releases as well as higher customer fulfillment. Software modules are categorized into two categories, either defective or non-defective, and are mostly predicted utilizing a binary classification model. We take advantage of these two classes for suggestions on how to classify and evaluate data sets.

Cagatay Catal [22] studied various papers in year 1990 to 2009 those are as following: they used classification trees with method level metrics on two software systems of NASA and Hughes Aircraft and also applied logistic regression, classification trees. Evett et al. predicted quality based on genetic programming system. They applied fuzzy subtractive clustering method to predict the number of faults and then, they used different module order modeling to classify the modules into faulty or non-faulty classes. They stated that process metrics is not improving the classification accuracy and such a model does not provide acceptable results. They used principal component analysis for first step that is feature selection and then applied fuzzy nonlinear regression to predict defects on a large telecommunications system developed with Protel language. They reported that fuzzy nonlinear regression method is an encouraging technology for early defect prediction. They observed that support vector machine performed better than quadratic discriminate analysis and classification tree. They focused on the high-performance defect predictors based on machine learning such as Random Forests algorithms.

Ezgi Erturk et al. [27] proposed a new method Adaptive Neuron Fuzzy Inference System (ANFIS) for the software fault prediction. Then for performing experiment they used PROMISE Software Engineering Repository dataset, and McCabe metrics are selected because they comprehensively address the programming effort. The results achieved were 0.7795, 0.8685, and 0.8573 for the SVM, ANN and ANFIS methods, respectively. Mie Thet Thwin [26] have used two kinds of neural network techniques. The first one focuses on predicting the number of defects in a class and the second on predicting the number of lines changed per class. Two neural network models are used which are: Ward neural network and General Regression neural network (GRNN). They have performed the analysis result on the NASA dataset. David Gray et al. have focused on classification analysis rather than classification performance, it was decided to classify the training data rather than having some form of tester set. It involves a manual analysis of the predictions made by Support vector machine classifiers using data from the NASA Metrics Data Program repository. Ensemble classifier also gives better result for classifying software defects [24]. The purpose was to gain insight into how the classifiers were separating the training data. Ruchika Malhotra [25] have analyzed and compared the statistical and six machine learning methods for software fault prediction. These methods (Decision Tree, Artificial Neural Network, Cascade Correlation Network, Support Vector Machine, Group Method of Data Handling, and Gene Expression Programming) are empirically validated to find the relationship between the static code metrics and the defects occurs in a module. They compared the models predicted using the regression and the machine learning methods. They have used two publicly available data sets AR1 and AR6 and among them decision tree gives best prediction result. Ahmet Okutan [29] have used Bayesian networks to determine the probabilistic influential relationships among software metrics and defect proneness. The software metrics used in Promise data repository, define two more metrics, i.e. number of developers for the number of developers and lack of coding quality for the source code quality.

Alina Campan et al.[28] they proposed a novel algorithm for the discovery of interesting any length of ordinal association rules in defect data sets. Datasets that contain several software metrics with similar or comparable domains of values are frequent in data mining. Gabriela Czibula et al. they proposed a supervised method for detecting software entities with architectural defects, based on relational association rule mining. They performed eexperiments on open source software are cconducted in order to detect defective classes in object oriented software systems for example the WinRun4J is a windows native launcher for Java implementation. Qinbao Song et al. [31] they calculate defect association, defect isolation effort, defect correction effort on SEL defect data consisting of more than 200 projects over more than 15 years. They compared the defect correction effort prediction method with other types of methods like PART, C4.5, and Naive Bayes and show that accuracy has been improved by at least 23 percent. They have explored the impact of support and confidence levels on prediction accuracy, false negative rate, false positive rate, and the number of rules as well. They found that higher support and confidence levels may not result in higher prediction accuracy, and a sufficient number of rules is a precondition for high prediction accuracy.

Kamei et al.[23] proposed a defect prone module prediction method that combines association rule mining with logistic regression. They have predicted performance of their algorithm method with different thresholds of each rule interestingness measure support, confidence and lift using a module set in the Eclipse project. Yuan Jiang, Ming Li et al.[9] have addressed two practical issues first, it is rather difficult to collect a large amount of labeled training data for learning a well-performing model and second, in a software system there are usually much less defective modules than defect free modules, therefore learning techniques would have to be conducted over an imbalanced data set therefore they proposing a novel semi-supervised learning approach named Random Committee with Under Sampling (Rocus). This method incorporates recent advances in disagreement-based semi-supervised learning with under-sampling strategy for imbalanced data. Above approaches have not used hybrid approach that is k-means clustering with Apriori algorithm for generating accurate rules regarding, they just focused on the relation association rule. This work focuses on improving performance of rule generation for software defect prediction. As in original work Apriori algorithm is used, it returns a large amount of results. Applying K-means algorithm in preprocessing step on results of defect prediction improve accuracy.

Above approaches have not used hybrid approach that is k-means clustering with Apriori algorithm for generating accurate rules regarding, they just focused on the relation association rule. This work focuses on improving performance of rule generation for software defect prediction. As in original work Apriori algorithm is used, it returns a large amount of results. Applying K-means algorithm in preprocessing step on results of defect prediction improve accuracy.

III. Conclusion

Software defect prediction work focuses on the number of defects remaining in a software system. The software defect prediction model helps in early detection of defects and contributes to their efficient removal and producing a quality software system based on several metrics. A prediction of the number of remaining defects in an inspected are fact can be used for decision making. An accurate prediction of the number of defects in a software product during system testing contributes not only to the management of the system testing process but also to the estimation of the product's required maintenance. Defective software modules cause software failures, increase development and maintenance costs, and decrease customer satisfaction. It strives to improve software quality and testing efficiency by constructing predictive models from code attributes to enable a timely identification of fault-prone modules. The main objective of this study was to assess the previous research works with respect to software defect which applies machine learning method, data set used, tools they used, methodologies, their contribution to science and we classified it in to three such as based on classification, Clustering and ensemble methods. Finally, it is possible to extend this study by systematic literature review which includes books, dissertation, tutorial, Thesis.

References

[1] S Bibi, G Tsoumakas, I Stamelos, and I Vlahavas. Software defect prediction using regression via classi_cation. 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] Iker Gondra. Applying machine learning to software fault-proneness prediction. Journal of Systems and Software, 81(2):186{195, 2008.

[4] Atac 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.

[5] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction",ACM/IEEE 30th International Conference on Software Engineering,ICSE'08, pages 181-190, 2008.

[6] H. Zhang, X. Zhang, and Ming Gu, "Predicting defective software components from code complexity measures", IEEE In Dependable Computing Pacific Rim International Symposium on, pages 93-96, 2007. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[7] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings",IEEE Transactions on Software Engineering, 34(4):485-496, 2008.

[8] R Geoff Dromey, "Software Control Quality - Prevention Verses Cure?", ACM Journal of Software Quality Journal archive, Volume-11 Issue 3, Pages 197-210, July 2003.

[9] Kaur S, Kumar D, "Software fault prediction in object-oriented software systems using density based clustering approach", International Journal of Research in Engineering and Technology (IJRET), 1(2):111-7, Mar-2012.

[10] Q. Song, Z. Jia, M. Shepperd, Shi Ying, and Jin Liu, "A general software defect-proneness prediction framework", Software Engineering, IEEE Transactions on, 37(3):356-370, 2011.

[11] Haghighi, A. A. S., Dezfuli, M. A., and Fakhrahmad, S. M., "Applying mining schemes to software fault prediction: A proposed approach

aimed at test cost reduction", In Proceedings of the World Congress on Engineering, pp.415-419, 2012.

[12] Software Defect Dataset, PROMISE REPOSITORY, http://promise.site.uottawa.ca/ SERepository/ datasetspage.html, December 4, 2013.

[13] H. Najadat and I. Alsmadi,"Enhance RuleBased Detection for Software Fault-Prone Modules", International Journal of Software Engineering and Its Applications,Vol. 6, No. 1, January 2012

[14] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering", IEEE Trans. Softw. Eng., 38(6):1276– 1304, Nov. 2012.

[15] Shepperd, M., Song, Q., Sun, Z., and Mair, C., "Data Quality: Some Comments on the NASA Software Defect Data Sets", IEEE Transactions on Software Engineering, pp.1208-1215, 2013.

[16] Okutan O. T. Yildiz, "Software defect prediction using Bayesian networks",Inproceeding to Empirical Software Engineering, pp. 1-28, 2012.

[17] H. Can, X. Jianchun, Z. R. L. Juelong, Y. Quiliang and X. Liqiang, "A new model for software defect prediction using particle swarm optimization and support vector machine", IEEE 25th Chinese Control and Decision Conference (CCDC), 2013.

[18] J. Wang, B. Shen and Y. Chen, "Compressed C4.5 Models for Software Defect Prediction",IEEE 12th International Conference on Quality Software (QSIC), pp. 13-16, August2012.

[19] E. Arisholm, L. C. Briand, and E. B. Johannessen. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. J. Syst. Softw., 83(1):2–17, Jan. 2010.

[20] T. GalinacGrbac, P. Runeson, and D. Huljenic, "A second replicated quantitative analysis of fault distributions in complex software systems", IEEE Trans. Softw. Eng., 39(4):462– 476, Apr. 2013.

[21] D. Rodriguez, I. Herraiz, and R. Harrison, "On software engineering repositories and their open problems", In Proceedings of RAISE '12, pages 52–56, 2012.

[22] C. Catal, Software fault prediction: "A literature review and current trends, Expert systems with applications", vol. 38, no. 4, pp. 4626-4636, 2011.

[23] Y. Kamei, A. Monden, S. Morisaki, and K.-i. Matsumoto, "A hybrid faulty module prediction using association rule mining and logistic regression analysis", in Proceedings of the Second ACM-IEE international symposium on Empirical software engineering and measurement, pp. 279-281, ACM, 2008..

[24] Tao WANG, Weihua LI, Haobin SHI, Zun LIU, "Software Defect Prediction on Classifier Ensemble", Journal of Information & Computational Sciences, 8:16(2011) 4241-4254.

[25] R. Malhotra, "Comparative analysis of statistical and machine learning methods for predicting faulty modules,"Applied Soft Computing, vol. 21pp. 286-297 2014.

[26] M. M. T. Thwin and T.-S. Quah, "Application of neural networks for software quality prediction using objectoriented metrics,"Journal of systems and software, vol. 76, no. 2, pp. 147-156, 2005

[27] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," Expert Systems with Applications, 2014.

[28] Campan, G. Serban, T. M. Truta, and A. Marcus, "An algorithm for the discovery of arbitrary length ordinal association rules.," DMIN, vol. 6, pp. 107-113, 2006.

[29] Okutan, Ahmet, and Olcay Taner Yıldız. "Software defect prediction using Bayesian networks." Empirical Software Engineering 19.1 (2014) 154-181

[30] .J. Nam, Survey on software defect prediction," 2010.

[31] Qinbao Song, Martin Shepperd, Michelle Cartwright, Carolyn Mair, "Software Defect Association Mining and Defect Correction Effort Prediction" , IEEE transaction on software engineering VOL. 32, NO. 2, Feb 2006