IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Volume-4, Issue-12 E-ISSN: 2347-2693

Prediction of Bugs in Software Repositories

S. Gomathi[1] and L. Haldurai[2*]

[1]Department of Computer Science, Kongunadu Arts and Science College, Coimbatore, India
[2*] Department of Computer Science (PG), Kongunadu Arts and Science College, Coimbatore, India

e-mail: gomathi.selvaraj93@gmail.com, haldurai@gmail.com

Available online at: www.ijcseonline.org

Received: 21/Nov/2016 Revised: 01/Dec/2016 Accepted: 14/Dec/2016 Published: 31/Dec/2016

***Abstract*— Defective software modules can leads to ad hoc software failures, shoots up development & maintenance cost and result in customer dissatisfaction. Defect mapping and awareness of its impact in different business applications paves way to improve its quality. Previous researches show that it has treated all bugs alike. Proper Identification and categorization helps to handle and fix bugs diligently. Evaluation of prediction techniques is mainly based on precision and recall measures. It focuses on the defects in a software system. A prediction of the number of left-out defects in an inspected arte fact can be judiciously used for decision making. An accurate prediction of quantum of defects during testing a software product contributes not only to manage the system testing process but also to estimate its required maintenance. It goes a long way to improve software quality and testing efficiency by building predictive models from code attributes to timely identification of fault-prone modules. In short, this paper provides the prediction of bugs by using data mining techniques such as Association Mining, Classification and Clustering. This complements developers to detect software defects and debug them. Unsupervised techniques come handy for defect prediction in software modules, on a large scale in those cases where defect labels are not present.**

Keywords- Software Defect Prediction, Bugs, Software Repositories, Data Mining, Classification, Clustering, Association Mining

I. INTRODUCTION

In recent years, there is increasingly a dramatic attention in reporting incidences of bugs resulting out from software applications. This is considered as an important and invaluable source for application's memory. Past errors play a pivotal role to carve future work of software applications. This is achieved through avoidance of same type of errors or accurately estimates the time and select good developers to solve upcoming issues. Many studies [1] show that more than 90% of the software development cost is lavished on maintenance and evaluation requirements. Software applications errors are managed and maintained in bug repositories or issue tracking systems [2].

Issue tracking system usually contains a knowledge base on each defect such as problem description, quick fix resolutions or impact analysis, project title, founder role, phase detected and phase injected. It is an open communication channel between multitude of people like end users, programmers and testers to find out the appropriate response about issues detected in software applications. Many reasons for software bugs include lack of awareness in requirements, dearth for good design in software application, difficulty in implementing applications and insufficient experience to code them. There can a plethora of defects types in projects. Some defects can lead to method failure while others can be deferred or missed such as spelling mistakes in error message [3].

*Corresponding Author: L. Haldurai
e-mail: haldurai@gmail.com

Defects are generally classified according to its impact on the functionality in application. An adept developer is assigned the responsibility to fix these errors with a stipulation in time to resolve issue within. For instance, security related issues / errors are complex in nature and require more time and more experienced developer to fix them which may not be required for performance related issues [4].

During emergence of mistakes with end users funneled via issue tracking systems, project manager will depute coordinators to assist them to rectify the mistakes. A Coordinator is one who selects the apt programmer and designer to investigate a problem. Investigator are a group of programmer / designer who find solutions to problems and estimate time to fix it as shown in Figure 1.

Coordinators have a fair experience in assessing the type of mistakes and assort them in precedence of importance. In mammoth projects, there are quite a large number of incoming defects noticed on a day-to-day basis. Coordinators main job is to focus on criteria of finding out defects nature and identify the correct person or action. This is really not an easy process. Assuming the errors reception rate in a system is 20-30 per day and the average time required to determine defect type is 10 minutes, then coordinator may typically take 3-5 hours to estimate the right type and then select proper destination for these defects. This above is from the perspective of time. But from practical front, in order to identify the defect type, coordinator have to necessarily view source code, system design and unit test documents to judge

© 2016, IJCSE All Rights Reserved

the right category of new born issues. They must also have domain expertise about business applied in project to judge error types. Assessment of defect nature is a manual process and lot of time and efforts is consumed in classifying bug types.

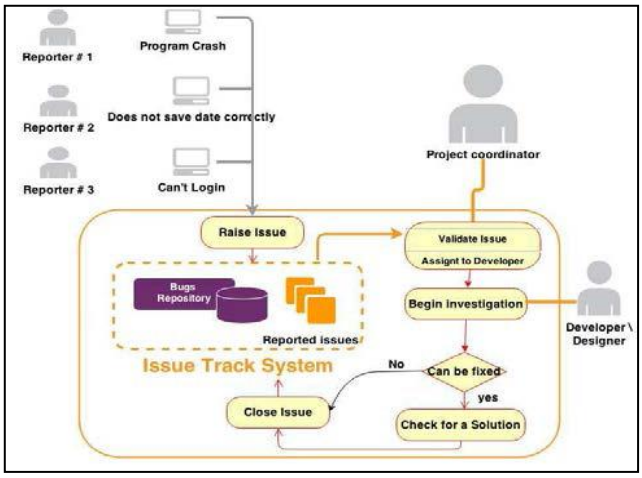

Figure 1. Bug Fixing Cycle [20]

II. RELATED WORK

In [5] built a model to detect defect correction effort based on extended association rule mining. They defined defect fixing effort as a variable and appropriate association rule mining to treat with such variables. Data used are supported from Japans Ministry of Economy, Trade and Industry (METI). They use support and confidence as evaluation factors. Their approach expressed results as a mean of correction effort based on development level.

In [6] predicted severity of bug report using classification model for severe and non-severe issues. They used online summary field of bug report for prediction based on SVM, Naive Bayes, Multinomial Nave Bayes and Nearest Neighbor Classifiers which make better performance in results. Results were evaluated using ROC (receiver operating characteristic) curve [7]. Performance of Multinomial Naive Bayes was found to be better than that of other classification algorithms.

In [8] developed an approach for predicting re-opened defects through Eclipse projects. Their study depend upon factors such as work habits dimension like: day which issue is closed, the bug report features dimension like: components, the bug correction dimension like: time needed to fix bug.

Analyzed bugs classes according to bug life time. He invented a model by sorting bugs with a shorter life time as a higher priority level [9].

Mining techniques were applied [10] on the bug report data to predict who should fix new coming bug. They used Support Vector Machines (SVM), Naive Bayes and Decision Trees algorithms on bug data of Bugzilla [11], Eclipse [12] projects.

In [13] analyzed the features of different types of bugs such as security and performance bugs to get useful information for their behavior in terms of the bug fix time, the number of developers assigned and the number of files impacted. Their Results show that security bugs are more complex, required more developers with experience, and large number of files affected but took less fix time than performance and other bugs. Similarly, performance bugs need more experienced developers than the other bugs.

Another important research about bug type detection is developed in [14] who proposed a text mining technique to determine security bug reports (SBR) from the set of undefined non-security bug reports (NSBR). A bug report's summary and long description fields were used for training the model. The bug data of Cisco software project was used to train model. The classifier is evaluated using the precision, recall and accuracy rate measures. Classifier is able to predict with percentage (78%) of SBRs that have been manually labeled as NSBRs. The research works on one kind of issues that are security and compared to manual selection of security issues.

From above discussion we did not find any related research on predicting bug category except Gegick,. They deal with security issues only that are selected manually and no analysis found in open source data sets like Bugzilla and Eclipse on defect category and impact analysis of bug reports. Bugzilla does not interest in adding impact analysis details on bug reports. Whereas Gegick's research select Cisco bug repository to reach impact analysis and identify bug classes.

III. ISSUE TRACKING SYSTEM

Issue tracking systems [15] is deployed to manage and maintain bugs list received from different actors in development life cycle. It works as a record for software application characteristics. It acts as connection channel between end users, developers, designers and testers. Communications are established through different activities like creating entirely new issues, reading existing issues, adding details to existing issues, or resolving an issue. When a user makes a change in the tracking system, all relevant data's like action and who made it etc., are the recordings which serves in maintaining a history of the actions taken. Each user of the tracking system may have issues assigned to him. He is responsible to find proper resolution to fix the issue. They have the option of reassigning an issue to another user, if needed. From security perspective, usually the

© 2016, IJCSE All Rights Reserved

tracking system authenticates its users before allowing access to the systems. Tracking system is a knowledge base containing information on each user, resolutions to common problems etc. Bug report is a major constituent of issue tracking. This is called a ticket. It is created from technical support team, development team or testing team as a result of an incident. By creating a ticket, a notification will be popped up to project manager or coordinator. Ticket has certain details about business scenario that causes unexpected behavior from software application.

In next two sections, let us describe bug report contents and bug life cycle.

A. Bug Report Contents

A major component of an issue tracking system is the Bug report. This has unique information on the incidents and a set of fields. Some fields describe incident in a natural way without any rules while some other fields brings in a set of predefined values. Description represents a 360 degree definition to regenerate the issue. Severity explains the criticality on how an issue will affect the software application. Additionally, it also carries a range of values such as critical / blocker, major, medium, minor and low. Founder role represents who has found the issue. It includes tester, end user, developer or analyst. "Phase detected" gives information as in which phase this issue is generated. Phase values depend upon organization development cycle that may include requirement, design, coding, function test and user acceptance test. "Phase Injected" represents in which phase issue originates. And lastly, impact analysis is the one which represents why this issue is generated and what are the changes needed to fix it. Defect category represents how this issue affects an application. It can be any one of a function, standard, graphical user interface or logic related.

B. Bug Life Cycle

Issue tracking systems have different states or phases or gates for a bug which can be best tracked through the status assigned to it. The moment an issue report is submitted, it gets a unique identifier by which it can be referred to in further communications. Was the issue gets processed, the report runs through a life cycle. The progress or the position in the life cycle is determined by the state of the issue report. Initially, every issue report will get a state of New Issue. The coordinator then checks its validity and uniqueness. One when these checks are passed / cleared, a developer / designer will be 'assigned'. It is important to identify if the issue is crucial to decide proper selection of developer and finding the resolution of issue. After this, Status of the report becomes 'Assigned'. At this point of time, the issue report is also assigned a priority. The higher the priority, the sooner it is going to be addressed. The developer now works on the issue; state of issue is changed to 'Under Investigation'. If developer finds a problem in source code, system design or application configuration and comes up with a resolution, then the status will be move 'Resolved', otherwise the issues will be 'closed'. At this stage, developer records the defect category. As the problem is now 'fixed', two more steps remain: the testers must confirm the success of the fix (resulting in under testing state). If the tester found issue is fixed, report will go to state 'Closed'. Otherwise issue will go back to 'Under Investigation' state.

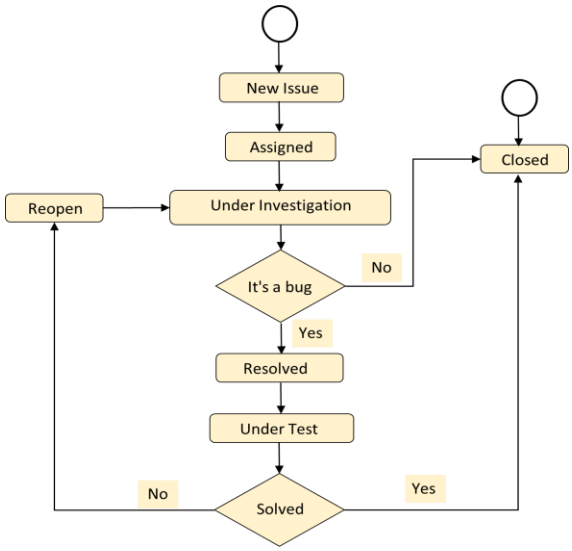

Figure 2. Bug Life Cycle [20]

IV. DATA MINING TECHNIQUES FOR DEFECT PREDICTION

Data for analysis is retrieved from software repositories. It has volumes of information that is useful in assessing the quality of software. Data mining techniques and can be applied on these repositories to extract the defects of a software product.

A. Clustering

Clustering is a type of unsupervised learning in which class labels are not provided. Clustering is the first data mining task deployed on a collection of data records that are grouped based on their similarity. In other words, Clustering is the task of organizing data into groups of similar nature and putting them into same cluster group. The groups are not predefined and hence clustering is deployed to partition data in a set of meaningful sub-classes. Clusters are subsets of objects that are similar. Clustering helps end users to understand the natural grouping or structure in a data set. Its schemes are evaluated based on the similarity of objects

© 2016, IJCSE All Rights Reserved

within each clusters. This approach offers benefits to experts who must decide the labels. Instead of inspecting and labelling software modules one at a time, the expert can inspect and label a given cluster as a whole; he or she can assign all the modules in the cluster to the same quality label.

Software Defect Prediction Using Clustering

In [16], [17], k-mean technique of clustering has been used for Software Defect Prediction. K-mean clustering is a non-hierarchical clustering procedure in which items are moved amongst sets of clusters until the desired set is reached. It has certain drawbacks. To overcome these drawbacks, Quad Tree-based k-mean clustering method was proposed. The objectives: first, Quad-trees are applied to finding initial cluster centres for k-mean algorithm. Second, the Quad tree-based algorithm is applied for predicting faults in program modules. Quad tree-based k-mean clustering algorithm was evaluated in comparison to the original k-mean algorithm for predicting faulty software modules. The result suggests that the number of iterations of k-means algorithm is less in case of Quad tree-based k-mean as well as percent errors were in fairly acceptable limits.

B. Classification

Classification is defined as a process of finding a set of models which describes and distinguishes data classes or concepts. It is the organization of data in given classes (also known as supervised learning) where the class labels of some training samples are given. These samples are invariably used to supervise the learning of a classification model. Classification approaches normally use a *training set* where all objects are already associated with known class labels. The classification algorithm learns from the training set and builds a model. The model is used to classify new objects. Fraud detection and credit risk applications are particularly well suited to this type of analysis. This approach frequently employs decision tree or neural network-based classification algorithms. The data classification process involves learning and classification. In Learning, the training data are analyzed by classification algorithm. In classification, test data are used to estimate the accuracy of the classification rules.

.

Software Defect Prediction Using Classification

Innumerable classification methods have been suggested to build software defect prediction models. In [18], an association rule classification method is proposed to derive a comprehensible rule set from the data. They have compared CBA2 [19] with other rule based classification method to check if classification algorithms based on association rules are suitable for software fault prediction. Studies has also been made to find out whether rule sets learned on one data set are applicable to others data sets. They have thoroughly investigated the performance of an association rule based classification method for software defect prediction. Experiments were conducted, results were compared with other classifiers and finally concluded that results were satisfying the performance requirements without losing comprehensibility.

C. Association Mining

Association mining task consists of a series of activities - identifying the frequent item sets**,** forming conditional implication rules etc. It is the task of finding correlations between items in data sets. Association Rule algorithms must be able to generate rules with confidence values less than one. Association rule mining is undirected or unsupervised data mining over variable-length data and it produces clear, understandable results. Association rules mining consist of two steps. First step involves the finding of the set of all frequent item sets. The second step involves the testing and generating all high confidence rules among item sets. It has a simple problem statement, that is, to find the set of all subsets of items that occur frequently in database records or transactions, and to extract the rules telling us how a subset of items influences the presence of another subset.

Software Defect Prediction Using Association Mining

In association rule mining technique we use defect type data to predict software defect associations that have relation among different defect types. The defect associations can be used for three purposes: Firstly, to find as many related defects as possible to detect defects and make more effective corrections to the software. Secondly, it helps to evaluate the reviewer's results during an inspection. Thirdly, it helps in assisting managers to improvise the software process through analysis the reasons why some defects frequently occur together. Association rule mining aims at discovering the patterns of co-occurrences of the attributes in the database. Results of analysis show that this technique does not yield higher support and higher confidence levels leading to lesser prediction accuracy.

V. CONCLUSION

Software defect prediction is the process of tracing defective components in software prior to the start of testing phase. Occurrence of defects is unavoidable, but we should try to limit these defects to minimum count. Defect prediction leads to reduce the time of development, cost, rework and it increases the customer satisfaction and reliability of software. Therefore, defect prediction is an important activity to achieve software quality and to learn from past mistakes. This paper elucidates the application of various data mining techniques for conquering software defects among existing repositories. To precise it predicts the availability of defects and is categorized.

© 2016, IJCSE All Rights Reserved

REFERENCES

[1] Erlikh.l, "Leveraging Legacy System Dollars for e-business", IT Professional, Volume-02, Page No (21 -22), March 2000.

[2] Joel spolsky, "Painless Bug Tracking", IT Professional, November 2000.

[3] Allen., mitch., "Bug Tracking Basics: A beginner's guide to reporting and tracking defects", The Software Testing & Quality Engineering Magazine. Volume- 4, Issue 3, Page No. (20–24), June 2002.

[4] Zaman S., Adams B., Hassan A.E., "Security versus performance bugs: A case study on Firefox". In Proceedings of the 8th Working Conference on Mining Soft-ware Repositories, May 2011.

[5] Shuji M., Akito M., Tomoko M ., " Defect data analysis based on extended Association Rule Mining", In Proceedings of International Workshop on Mining Software Repositories (MSR), 2007.

[6] Lamkanfi A., Demeyer S., Soetens Q D., Ver- donck T., "Comparing mining algorithms for predicting the severity of a reported bug". In Proceedings of the 15th European Conference on Software Maintenance and Reengineering, March 2011.

[7] Ling C., Huang J., Zhang H., "AUC: A better measure than accuracy in comparing learning algorithms", In Lecture Notes in Computer Science Page No. (26–71), Springer-Verlag, 2003.

[8] Emad S., Akinori I., Walid I., Ahmed H., "Predicting reopened bugs: A case study on the eclipse project". In Proceedings of the 17th Working Conference on Reverse Engineering (WCRE) 2010.

[9] Kim S., Ernst M D., "Prioritizing warning categories by analyzing software history". In Proceedings of the 4th Inter-national Work shop on Mining Software Repositories, Minneapolis, USA, May 2007.

[10] Anvik J., Hiew L., Murphy G C., "Who should fix this bug?" In Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, May 2006.

[11] Mozilla Products track system, https://bugzilla.mozilla.org/

[12] Eclipse Bug repository https://bugs.eclipse.org/

[13] Zaman S., Adams B., Hassan A E., "Security versus performance bugs: A case study on Firefox". In Proceedings of The 8th Working Conference on Mining Software Repositories, May 2011.

[14] Gegick M., Rotella P., Xie T., "Identifying security bug reports via text mining: An industrial case study". In Proceeding of the 7th Working Conference on Mining Software Repositories, May 2010.

[15] Joel Spolsky, "Bug Tracking System Definition", Empirical Software Engineering, November 08, 2000.

[16] Partha Sarathi Bishnu., Vandana Bhattacherjee., "Software Fault Prediction Using Quad Tree-Based K-Means Clustering Algorithm", IEEE: International Conference on Transactions on knowledge and data engineering, Volume- 24, Page No. 6, June 2012.

[17] Qinbao Song., Martin Shepperd., Michelle Cartwright., Carolyn Mair., "Software Defect Association Mining and Defect Correction Effort Prediction", IEEE: International Conference on Transactions on Software Engineering, Volume- 32, Page No. 2, February 2006.

[18] Baojun Ma., Karel Dejaeger., Jan Vanthienen., Bart Baesens., "Software Defect Prediction Based on Association Rule Classification", The 2010 International Conference on E-Business Intelligence.

[19] Liu B., Ma Y., Wong C K., "Improving an association rule based classifier," In Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases, 2000.

[20] Mostafa M. Ahmed ., Abdel Rahman M., Hedar., Hosny M.Ibrahim., "Predicting Bug Category Based on Analysis of Software Repositories", 2nd International Conference on Research in Science, Engineering and Technology(ICRSET'2014), Page No. (21-22) March 2014 Dubai.

Authors Profile

Ms. S. Gomathi had done her Bachelor of Science and Master of Science from Bharathiar University. She currently pursues Master of Philosophy in Computer Science at Kongunadu Arts and Science College. Her main research work focuses on Data Mining and Software Engineering.

Mr. L. Haldurai had done his Bachelor of Science, Master of Computer Applications and Master of Philosophy from Bharathiar University. He currently works as Assistant Professor in Department of Computer Science (PG), Kongunadu Arts and Science College. His main research work focuses on Data Mining and Data Communications. He has 8 years of Teaching experience and 2 years of Research experience.

© 2016, IJCSE All Rights Reserved