A SURVEY ON PREDICTION OF PRIORITY AND SEVERITY IN THE BUG REPORTS

Pushpalatha MN[1], Deeksha DR[2]

[1] *Assistant Professor, Department of Information Science and Engineering,*

Ramaiah Institute of Technology, Karnataka, India

[2]*Student, Department of Information Science and Engineering,*

Ramaiah Institute of Technology, Karnataka, India

ABSTRACT

Software usage has increased by leaps and bounds and it is being used in every sphere of human activity. Bugs in the software would make it ineffective or unstable. As per the statistics 57% of the bugs originate through human made errors. This in turn results in cost overruns, delayed timelines and higher maintenance costs. Detecting the bugs is very crucial at the initial phase as it affects both the quality and accuracy after its deployment. In order to report these kinds of bugs, a system termed "Bug Tracking System" (BTS) exists. While testing any software, BTS receives a huge number of bugs on a daily basis. If the triage (the one who uses his knowledge and experience to analyze and refine the reported bug) starts reading all the bugs then it is possible that many important bugs will remain untreated and it is also difficult for the triage to resolve the bugs within the available time and resources. Hence, Bug priority and severity is an inevitable process in the software maintenance.

This paper emphasizes in necessitating a short and fleeting survey on numerous bug priority and severity techniques. We are reviewing and categorizing survey of 30 literature research works in the domain of bug priority severity.

Keyword: - *Priority and severity in bugs; concerns*

1. INTRODUCTION

Large spheres of human activities are influenced by the software and the software application is increasing at an incredible rate. Due to increased demand, ensuring quality with decreased delivery time has become very critical, i.e. shorter delivery time has become the utmost requirement and hence there is a need to ensure that the quality is maintained during the process. To ensure the quality in software different methods have been adapted such as persistent, attentive testing and code reviews so as to remove the bugs in the early phases thereby preventing the loss. Going on with an ancient saying, "Every software program is never perfect, there is at least one bug in it which can be encountered at any time"[25].The bugs can be responsible for the software acting unrelated from its specification, which may encounter in the course of product operation either in usage or under the test.

Mistakes that occur due to the human involvement and participation are termed as bugs. As per a Statistics 57% of the bugs originate through human made errors. This could happen because of carelessness or absent mindedness. Such irresponsible causes may results in software failures which implicitly or unreservedly cost's companies a randomized amount of money and also in loss of human lives will also occur in some instances.

Eg1: Software bug in a Royal Air Force Chinook aircraft's engine control computer resulted in a crash in 1994 and 29 people lost their lives because of this incidence [25].

Therefore detecting bugs are very crucial at the initial phase. The bugs detected after software deployment disturbs both quality and accuracy of the software. In order to report these kinds of bugs, a system exists termed as Bug Tracking Systems (BTS). A BTS recognizes and thus provides the solution for the bugs which are reported henceforth also refining complete quality of the generated software. A bug can be reported by any developer, user or tester. Bugs address by the BTS will be initially scrutinized and examined by Triage so as to estimate their legitimacy, perfection, prominence and also to authenticate its duplicity. After which these bugs will be assigned to related developers to resolve the same. The person who implies his ability, familiarity in analyzing and also refining of bugs which are reported is termed as Triage and the process is defined as Bug Triaging Process.
Physically allocating the priority and severity occasionally may go wrong due to the inexpert users or operators in open source software. In case of profitable software development, test engineers will be responsible in severity allocation. If in case the test engineer is a fresher and has inadequate experience, then it is a good practice for him/her in utilizing reference organizations in assigning the severity levels.
In other cases, the recommendation system can be used to assign the severity or priority level when the test engineer is occupied with another task and this can save the time as manual process needs more time.

2. OVERVIEW OF BUG PRIORITY AND SEVERITY

Software developers dedicate foremost percentage of the existing resources in handling the bug reports produced by the end users. For the software's which are comprehensively organized, amount of bug reports typically outperforms resources accessible to triage them. On the basis of priorities, certain bug reports will be handled deliberately. Assuming if the developer spares five minutes to recite and grip on every report, two person-hours per day need to be disbursed for this activity. To cut off the time expended on triaging is the vital purpose to automate the system. Users are requested to initially rate bug priority and severity found through bug tracking system. Severity of the bug is bestowed by the users and the developers at that point deliver priority to the reports grounded on the severity.

Major problem in indicating priority and severity to the bug is that

- The bug may not be identified properly to which category it belongs to due to unqualified reporter.
- The report indicates the category of the priority and hence the developer should ascertain the exact differences among the category which most of the developers unable to cope with all the bugs that are listed in the report.

2.1 Bug Priority and Severity levels

All through the bug triaging process, a software development team decides by what means and how swiftly a bug needs to be fixed, using five sort priority level schemes P1, P2, P3, P4 and P5. Bugzilla also has the same priority levels P1 indicates the highest priority and these priority bugs should be fixed before launching the product whereas P5 is the lowest priority that can remain unfixed for a longer period [31].

P1 - critical, needs immediate attention
P2 - Important but can be resolved before the next launch
P3 - Not critical and can be delayed
P4 - Least priority and does not have any implication on the results
P5- Fixing can be deferred until all other priority defects are fixed

In the same line of the priority levels the approaches for assigning the severity labels are also followed. All the previous studies suggest many approaches for allotting severity labels for bug reports. The key approach behind all this study is that severity label of the newly reported bug report can be found using properties land from old bug reports where the severity label are already assigned.
The description and the summary field of the bug report give the brief ideas of the property of the report. For instance the reporter of the bug report can use the key words like 'critical', 'major' and 'blocker' to indicate top severity reports. On the other hand words like 'minor', 'trivial', 'enhancement' indicate that the bug report is of low severity [32].

Based on severity level the bugs are categorized into trivial, enhancement, minor, normal, major, critical, and blocker, the severity levels are briefed below:

- **Block**er: It blocks further development and also hinders the testing work
- **Critical:** It is applied for the bug which may results malfunctioning of all application
- **Major:** It is applied for the bug which may cause major loss of functionality for most of the applications
- **Minor:** For minor loss of functionality, or other problem where an easy work around is present
- **Trivial:** Intended for a UI glitch that doesn't interrupt program working
- **Normal:** Default case
- **Enhancement:** Enhancement request

3. LITERATURE REVIEW

This section focuses on the literature review carried out in the field of prediction of priority and severity involved in the bug reports.

Podgurski et al., [1] have proposed automated support to classify the software failures reported and help out in diagnosing the cause. The classification strategy is been designed and developed using supervised and unsupervised pattern classification and multivariate visualization. Their experimental results showed the efficiency of the usage of classification strategy.

Murphy et al., [2] have applied machine learning technique for assisting bug triage using text categorization in prediction for the developer with respect to bug's description. Authors have demonstrated their proposed work on 15,859 bug reports collected from open source project. The results showed that, supervised Bayesian learning can precisely predict to around 30% of the report assignment.

Anvik et al., [3] have proposed semi-automated approach for the assignment of the reports to the developer. They have used machine learning based algorithm to the open bug repository to learn the different kinds of reports each developer resolves. Authors have reached precision levels of 57% and 64% on the Eclipse and Firefox project by using the proposed technique and their technique has also applied on GCC open source development.

Menzies et al., [4] have developed an automated method called SEVERIS (SEVERity ISsue assessment) for hindering the test engineer in allocating severity levels for defect reports. Their proposed approaches used text mining and machine learning approaches applied on the conventional existing defect reports. Authors has also presented case study of the proposed method by using datasets from NASA's Project and Issue Tracking System (PITS).

Jeong et al., [5] have introduced graph model based method using Markov chains for capturing bug tossing history. Authors tested their method on 445,000 bug reports and results obtained showed 72% reduction in the tossing events and has enhanced the accuracy of bug assignment up to 23%.

Bhattacharya et.al., [6] have designed several techniques like refined classification using attributes and intra-fold updates during training to improvise the triaging accuracy and to reduce tossing path lengths. Authors have validated their technique covering 856,259 bug reports on two large software projects like Mozilla and Eclipse. Their work concluded that, prediction accuracy in bug triaging can be achieved up to 83.62% by using this proposed method.

Jianhong et al., [7] have written the paper on comparative study of five Neural Network Based techniques namely Batch Gradient Descent, Batch Gradient Descent with momentum, Variable Learning Rate, Variable Learning Rate training with momentum and Resilient Back-propagation for the modeling of severity of faults present in function based software systems. Dataset used for the study was from NASA's public domain defect. Their work concluded that Resilient Backpropagation algorithm based Neural Network is best compared to other techniques for modeling of software components.

Kanwal et at., [8] have proposed a technique based on machine learning approach for automatically handling the appropriate priority level for newly arrived bugs. Their work contributed in resolving bug in order without leaving it untreated for a long time using support vector machine based on classification technique. The experimental results have reported feasibility of the proposed approach in automatic detection of bug priority assignment.

Lamkanfi et al., [9] have proposed a technique to learn the prediction of severity of reported bug. The authors have demonstrated novel approach based on text mining algorithm for the accurate identification of the reported bug. Authors have tested there algorithm on the three different cases taken from the open source community. Authors have concluded that the prediction of severity could provide better accuracy based on the selection of training data set with the sufficient size.

Yu et al., [10] have developed a neural network technique for predicting the priorities of defects. Their work adopted evolutionary training process for solving the error problems linked with newly obtained features and reused the data sets from the related software systems for the faster convergence of training. Authors work provided the major framework for the model evaluation to demonstrate the feasibility and effectiveness.

Chen et al., [11] have developed the technique based on bug tossing history and textual comparison between bug reports for achieving quick identification of fixer to bug reports. Proposed work was evaluated on Eclipse and Mozilla and results showed the efficiency of proposed technique by significant improvisation of bug assignment.

Park et al., [12] have addressed two direction of bug fixing process by combining the existing Content-Based Recommendation (CBR) and collaborative filtering recommender (CF) for enhancing the quality. Due to the nature of bug fixes, one bug is fixed by only one developer, which makes it challenging to pursue the above two directions. To address this challenge authors have developed topic-model for reducing sparseness to enhance the quality of Content-Boosted Collaborative Filtering (CBCF).Their experimental results showed cut down in the cost efficiency by 30% without compromising accuracy.

Tamrawi et al., [13] have proposed novel approach called Bugzie for automatic bug triaging using fuzzy set and cache-based model. Author's experimental result concluded that Bugzie approach provides higher accuracy and efficiency compared to conventional state-of-art approaches.

Tamrawi et al., [14] have proposed novel approach called Bugzie for automatic bug triaging using fuzzy set based modeling. Author's experimental result concluded that Bugzie approach provides higher accuracy and efficiency compared to conventional state-of-art approaches.

Zou et al., [15] have developed training set reduction technique using feature selection and instance selection for bug triage. Authors had combined feature selection with instance selection in order to improvise the accuracy of bug triage. Training set reduction on the bug data of Eclipse has been evaluated. Authors have concluded that efficiency of the results can be improved by selecting new and small training sets compared to original one.

Chaturvedi et al., [16] have written a paper on applicability of machine learning algorithm used in finding the class of bug severity of the bug report data of NASA from PROMISE repository. Application of different algorithm for determining various levels of bug severity has been validated using 5-fold cross validation.

Kanwal et al., [17] have proposed classification based technique for handling out the new bug reports identified. Classifiers like Naïve Bayes and Support Vector Machine (SVM) classifiers are used to evaluate the efficiency of the classifier with respect to the accuracy. Authors used precision and recall measures for evaluating the bug priority recommender. Their development proposed to new measures namely Nearest False Negatives (NFN) and Nearest False Positives (NFP) for providing the insight to the results obtained from the precision and recall. Authors concluded that SVM method out perform for the text features whereas Naive Bayes algorithm performs well for categorical features.

Sharma et al., [18] have provided a comparative study for evaluating the performance of different machine learning techniques for prediction of priority of newly coming reports, keeping performance measures as the basis. Their experimental studies performed cross validation by choosing 5 data sets with 76 cases involved in open office and eclipse projects. Results obtained showed that prediction of priority of reported bug within and across project was more than 70% except Naive Bayes technique.

Tian et al., [19] have developed a leveraging information retrieval approach for automatically forecasting the severity included in the bug reports. Authors have used BM25 created document similarity function for achieving their desired results. In order to increase their prediction accuracy, they have investigated bug reports which were reported in the past and have classified newly reported bug reports by assigning new severity label to it. They also

have used duplicated bug reports for their studies in order to extract the features involved in their bug. Proposed method mainly focused on the predicting fine-grained severity labels. Their results concluded that proposed technique improves fine-grained severity prediction compared to the conventional existing techniques.

Xuan et al., [20] have investigated the problems associated with developer prioritization for providing better pathway for the developers. In this paper, Authors have mainly looked at two major aspects, one at modeling the developer prioritization in bug repository and other at supporting predictive tasks. Their work investigated at three major problems associated with developer rankings in products, the evolution over time and the tolerance of noisy comments. Results concluded that developer prioritization provides the insight of developer priorities for assisting the software tasks dealing with bug triage.

Zhang el al., [21] have designed and implemented automated developer recommendation tool for bug triage. Authors have contributed with respect to building the concept profile (CP) which helps out in extracting the bug concept. Evaluated results show that, their approach outperformed compared to other developer recommended methods.

Alenezi et al., [22] have developed automated approach to predict developer in solving the newly coming report in order to reduce time and cost related to bug triaging. Authors in their paper have evaluated the use of five term selection methods on accuracy of bug assignment. The experimental results obtained showed that selection of small number of discriminating terms can help in improvising the F-score.

Yang et al., [23] has developed a novel approach for the prediction of bug triage and bug severity. Authors used historical bug reports present in the bug repository for extracting the topics along with finding bug reports related to the each topic. Their approach used multi-feature for finding out corresponding reports which contains the same multi feature. They evaluated their approach using 30,000 golden bug reports chosen from the three open sources software.

Rana et al., [24] have proposed the novel method using feature selection and classification approach for classifying the bug reports into severe and non-severe classes. Authors work aimed in predicting bug severity automatically.

Kaur et al., [25] have introduced a technique for creating dictionary of critical terms in order to look at the bug severity as severe and non-severe. Authors in their work concluded that, usage of different approaches for feature selection and classification of pattern for accuracy and precision are same.

Pushpalatha et al., [26] the bug report severity is predicted using bagging ensemble method and it is also compared with C4.5 classifier (J48). Authors have concluded that, on a given datasets better accuracy is provided by bagging ensemble method than C4.5 general classifier.

Arudkar et al., [27] proposed a technique based on the random forest for the bug triage prediction. Authors used the proposed method for testing it on open source projects that generate huge bug data. Their method helped in data processing to reduce the scale by providing high quality bug data in software development. Experimental results concluded in the improvisation of accuracy in tracking system.

Xuan et al., [28] have proposed semi-supervised text classification technique for bug triage in order to avoid the shortage of labeled bug reports present in the existing supervised approaches. Their proposed design consists of combination of Naive Bayes classifier and expectation maximization approach. Author's result proves that, their new approach performs better than the existing supervised approaches.

Pushpalatha et al., [29] the severity is predicted using various ensemble techniques like Adaboost, Bagging, Random forest and Voting and concluded that bagging method gives accurate results when compared with other ensemble methods.

Umer et al., [30] have developed novel automatic approach using emotion-based in prediction of priority of the reports. Authors used natural language processing method for the pre-processing of bug report and also they identified the emotion-words which were involved in bug report. Proposed method was evaluated on Eclipse open source projects and results concluded that proposed method outperforms conventional state of the art while increasing the F1 score by 6% and more.

4. CONCLUSIONS

Once a bug tracking scheme accepts a newfangled filed bug report, the triage decides regarding numerous characteristics of bug reports such as priority and severity levels. The bug priority and severity level designates the prominence of that bug from commercial perspective. It gives the suggestion of the order in which bug reports ought to be fixed.

5. REFERENCES

[1] Podgurski Andy, David Leon, Patrick Francis, Wes Masri, Melinda Minch, Jiayang Sun, and Bin Wang "Automated support for classifying software failure reports" In 25th International Conference on Software Engineering, pp. 465-475, 2003.

[2] Gail C. Murphy, and Davor Cubranic "Automatic bug triage using text categorization" In Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering, pp.1–6, 2004.

[3] John Anvik, Lyndon Hiew, and Gail C. Murphy "Who should fix this bug?" In Proceedings 28th International Conference on Software Engineering, pp. 361-370, 2006.

[4] Tim Menzies, and Andrian Marcus "Automated severity assessment of software defect reports" In IEEE International Conference on Software Maintenance, pp. 346-355, 2008.

[5] Gaeul Jeong, Sunghun Kim, and Thomas Zimmermann "Improving bug triage with bug tossing graphs" In Proceedings of the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 111-120, 2009.

[6] Pamela Bhattacharya, and Iulian Neamtiu "Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging" In IEEE International Conference on Software Maintenance, pp. 1-10, 2010.

[7] Zhou Jianhong, Parvinder S. Sandhu, and Seema Rani "A Neural network based approach for modeling of severity of defects in function based software systems" In 2010 International Conference on Electronics and Information Engineering, vol. 2, pp. 568-575, 2010.

[8] Jaweria Kanwal, and Onaiza Maqbool "Managing open bug repositories through bug report prioritization using SVMs" In Proceedings of the 4th International Conference on Open-Source Systems and Technologies (ICOSST), pp.1-7, 2010.

[9] Ahmed Lamkanfi, Serge Demeyer, Emanuel Giger, and Bart Goethals "Predicting the severity of a reported bug" In 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), pp. 1-10, 2010.

[10] Lian Yu, Wei-Tek Tsai, Wei Zhao, and Fang Wu "Predicting defect priority based on neural networks" In International Conference on Advanced Data Mining and Applications, ADMA 2010, pp. 356-367, 2010.

[11] Liguo Chen, Xiaobo Wang, and Chao Liu "An Approach to Improving Bug Assignment with Bug Tossing Graphs and Bug Similarities" In Journal of Software, Vol. 6, No. 3, pp.421-427, 2011.

[12] Jin-woo Park, Mu-Woong Lee, Jinhan Kim, Seung-won Hwang, and Sunghun Kim "CosTriage: A cost-aware triage algorithm for bug reporting systems" In Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.

[13] Ahmed Tamrawi, Tung Thanh Nguyen, Jafar M. Al-Kofahi, and Tien N. Nguyen "Fuzzy set and cache-based approach for bug triaging" In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, pp. 365-375, 2011.

[14] Ahmed Tamrawi, Tung Thanh Nguyen, Jafar Al-Kofahi, and Tien N. Nguyen "Fuzzy set-based automatic bug triaging (NIER track)" In Proceedings of the 33rd International Conference on Software Engineering (ICSE)**,** pp. 884-887, 2011.

[15] Weiqin Zou, Yan Hu, Jifeng Xuan, and Jiang He "Towards training set reduction for bug triage" In Proceedings of the 35th Annual IEEE International Computer Software and Applications Conference, pp. 576-581, 2011.

[16] Krishna Kumar Chaturvedi, and V. B. Singh "Determining bug severity using machine learning techniques" In 2012 CSI Sixth International Conference on Software Engineering (CONSEG), pp. 1-6, 2012.

[17] Jaweria Kanwal, and Onaiza Maqbool "Bug prioritization to facilitate bug report triage" In Journal of Computer Science and Technology, pp. 397-412, 2012.

[18] Meera Sharma, Punam Bedi, K. K. Chaturvedi, and V. B. Singh. "Predicting the priority of a reported bug using machine learning techniques and cross project validation "In 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 539-545, 2012.

[19] Yuan Tian, David Lo, and Chengnian Sun "Information retrieval based nearest neighbor classification for fine-grained bug severity prediction" In 2012 19th Working Conference on Reverse Engineering, pp. 215-224, 2012.

[20] Jifeng Xuan, He Jiang, Zhilei Ren, and Weiqin Zou "Developer prioritization in bug repositories" In 2012 34th International Conference on Software Engineering (ICSE), pp. 25-35, 2012.

[21] Tao Zhang, and Byungjeong Lee "An automated bug triage approach: A concept profile and social network based developer recommendation" In International Conference on Intelligent Computing, pp. 505-512, 2012.

[22] Mamdouh Alenezi, Kenneth Magel, and Shadi Banitaan "Efficient bug triaging using text mining" In Journal of Software, Vol. 8, No. 9, pp.2185-2191, 2013.

[23] Geunseok Yang, Tao Zhang, and Byungjeong Lee "Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports" In 2014 IEEE 38th Annual Computer Software and Applications Conference (COMPSAC), pp. 97-106, 2014.

[24]Pankaj Rana, and Saurabh Sharma "Implementing bug severity prediction through information mining using KNN classifier" In International Journal of Science Technology & Engineering (IJSTE), Volume 2, Issue 04, pp. 333-340, 2015.

[25] Prabhsharan Kaur, and Charanjeet Singh "A Systematic approach for bug severity classification using machine learning's text mining techniques" In International Journal of Computer Science and Mobile Computing (IJCSMC),Vol. 5, Issue. 7, pp.523 – 528, 2016.

[26] M N Pushpalatha, and M Mrunalini "Predicting the severity of bug reports using classification algorithms" In International Conference on Circuits, Controls, Communications and Computing (I4C), pp. 1-4, 2016.

[27] Shreya Arudkar, and Amit Pimpalkar "Design of an effective mechanism for automated bug triage system" In International Journal of Research in Science & Engineering (IJRISE), Volume 3 Issue 1, pp. 298-304, 2017.

[28] Jifeng Xuan, He Jiang, Zhilei Ren, Jun Yan, and Zhongxuan Luo "Automatic bug triage using semi-supervised text classification" In Proceedings of the 22nd International Conference on Software Engineering & Knowledge Engineering, pp.209-214, 2017.

[29]M N Pushpalatha, M Mrunalini "Predicting the severity of closed source bug reports using ensemble methods" In Proceedings of the Second International Conference on Smart Intelligent Computing and Applications(SCI 2018), Volume 2, pp 589–597, 2018.

[30] Qasim Umer, Hui Liu, and Yasir Sultan "Emotion based automated priority prediction for bug reports" In IEEE Access, Volume 6, pp.35743-35752, 2018.

[31] Jamal Uddin, Rozaida Ghazali, Mustafa Mat Deris, Rashid Naseem, and Habib Shah "A survey on bug prioritization" In Artificial Intelligence Review, Volume 47, Issue 2, pp 145-180, 2017.

[32] Yuan Tian, Nasir Ali, David Lo, and Ahmed E. Hassan "On the unreliability of bug severity data" In Empirical Software Engineering, pp-2298-2323, 2015.