THE PERFORMANCE OF FUZZY LOGIC HOW TO FIXED THE RISK LEVEL OF BUG SYSTEM AS HARD-WORKING QUALITY SOFTWARE

Agus Pamuji
Informatic Engineering
IAIN Syekh Nurjat Cirebon
jurnal.agus.pamuji@gmail.com

Abstract— The quality of software production is considered important when testing which is involves several IT Staff such as IT development, operation, end-user. One of the issue was having today is a bug processing where it almost all platforms too difficult to avoid from the bugs and even might be full of the risks. In the main of Our focus is on how to measure and attempt to reduce the number were indicated as bugs from low up to critical levels. Furthermore, we were propose with a method already known as a fuzzy logic approach to measure the severity of the presence of bugs during the testing process. there are 20 thousand even more bugs have been reported and be supposed removed with the fuzzy logic approach with various levels. As The end result is that we have found a gradual 20% reduction in various criteria in the testing process as experimentally. Therefore, fuzzy logic is considered as effective enough to be able to improve existing methods and support to reduce bugs significantly.

Keywords—Bugs, Risk, Critical, Reduction, Quality.

I. INTRODUCTION

Currently, the quality of software is a lot of demand from every end-user as dominant participant [1]. The process of production a program into software needs to be debugged [2]. However, those process consumes a lot of time and effort that makes quality declining. The debug phase has been considered important as well as crucial because it has been widely studied and examined beforehand [3]. Developers, testers, and users have an important role during the debugging process [4], where as they communicate with each other team and coordinate in an effort to have the same perception. Besides that, the debugging process suppose reduce both in time [5]and costs [6]. In essence, the process of the bug can keep and stored a history of bug information as well as requests for software requirements to learn from various kinds of bugs through maintenance [7], or even during the development process [8]. Besides the bug process, the main concern is the quality of software where a software product is evaluated and tested based on the originality requirements of its users [9].

The bug is considered a software defect [10] and occurs in the development phase [11] even in software testing. Existing defects or bugs are main issue during the development process and were caused by mistakes and failures made by people, as well as program codes or error in the technical designs. On the other hand, a number of bugs indicate the poor quality of the software [12] since the bug caused the failure of software production to have an impact [13] on the experience of users who use it [14]. The bug process provides an opportunity for developers to improve software performance [15] or quality. In addition, in dealing is relatively difficult [16], [7], [17] to manage their bugs [18]. Tracking bugs also have a means tracking bugs reported in a software product [17]. Usually the bug process was generated through web based where there is a change request. Every single item added will be tracked, identified [19], and reported as a progress report [20]. Everyone needs to track all problems because they tend to forget and lag behind problems that exist in the system.

In the Our concern is how to measure and classification even we must to reduce and eliminate the risk of bugs which is their various or have high critically. In this paper, we would propose a framework in order to measure and reduce bugs, so that, it can affect the quality of software. In previous studies, it was more likely to build models, analyzes, and classification of tracking software defects but received little attention to how these models could be considered reliable. Therefore, we adopted a model that had previously been proposed and then we measured and classified it to support software quality. This paper can be organized consisting of an introduction that discusses the background, problem formulation, and purpose of the study. Furthermore, its followed by research methodology related to data retrieval and analysis, discussion of experimental results and conclusions added in the form of suggestions for future studies.

II. RELATED WORK

There are several approaches or methods in which it had been used in dealing with the bug problems or system bugs. Tao Zhang et. Al were proposed a technique applied in managing bug reports with bug-based procedures. The proposed technique allows to open feedback towards on system bugs in order to classify bugs. The purpose of this classification was marked by duplication of bugs which can be detected in the system so that it takes a lot of time when identifying bugs by some developers. In this framework, Tao Zhang proposes a 4c model that consisting of concepts, content, context and categories. The steps has applied as initial step with analysis and classification in textural using hierarchical classification algorithms. Second, the use of classification to facilitate debug tracking, and then use feedback from the developers. This phase was conducted which in identification of duplication indications, identification of validity and identification of classification of bug reports. But the results supposed need to be conducted in tracking bugs automatically because there are still might be many duplications that make a lot of spent time that considering the rise an effect on software quality. In the similiar content, it's actually how to detect duplication of bugs in the reported system bug. This study was conducted by Akihiro Tsuruda et. Al where it identifies its own bug life cycle. The method was proposed that the relationship between the number of words used to detect duplicate bug reports and the level of accuracy. Second, the relationship between the number of words used to detect duplicate bug reports and the number of bug reports classified into duplicate bug reports. There are four data sets used, namely Eclipse, Open Office, Mozilla, and Netbeans. Furthermore, it can be clarified that the more the number of words that are debugged does not have a lot of accurate effect on bug reports whereas the risk of duplication increases even though it initially decreases so that the results are quite significant but still have a relatively small risk of duplication.

Remember the several cases that indicate duplication of bugs during the development process or software testing, Ahmed Tamrawi et. Al proposed in his study. He proposed a method for automatically tracking bug duplications based on fuzzy rule logic. Initially it was proposed in the form of Bugzie. This technique is a novel approach. This Fuzzy method is able to overcome to become a relevant bug through the membership function extracted from bug reports. The final results of the study are quite significant even though there is still a chance for the risk of duplicating the reported bugs. However, it will be a challenge for researchers to deal with long-lived bugs that will frustrate many people , including users. Today, new long lived bugs can only be analyzed as reviewed by Ripon K. Saha et. Al. analysis was conducted by prioritizing in order to reduce the impact and also the number of emerging bugs is proven to have decreased by down-to 50%. This

condition might be still related to the bug and how the strategy was taken as proposed by Fei Pang et. Al. Bugs that emerge must be arranged and made strategies even though there are many methods and approaches that have been implemented. Fei Pang creates and proposed a framework by paying attention to bug workflows and can successfully effectively help programmers. This project involved 20 students by investigating the bugs were found. It should be emphasized that, in his study he used two objects besides focusing on bug workflows but also the flow of the bug data. Based on the results of the two comparisons, it turns out that the workflow of bugs is more efficient than the data flow bug that reaches a difference of 5%. Another approach related to bugs was proposed by Zatul Amilah et. Al where they made a bug tracking system. Tracking system designed to facilitate communication between developers, users, and others. Every problem is entered into the system and everyone involved can see the latest problem. But the system has not shown significant results regarding the process of remembering bugs affecting the quality of the software. This bug tracking system is also almost the same as Afreen Patel et. Al in 2013 is by creating a system with a web-based platform.

Defect tracking model in the classification of disability reports through tools (Torky, 2013). Starting from a literature review of the theory of differences in defect tracking models and previously trying to improve the defect tracking system. The result is a defect tracking model. Related to classification treatment was also reviewed by N. K. Nagwani et. Al by using the latent Dirichlet Allocation (LDA). The technique proposed is applied to open source Java and Mallet technology while the dataset is taken from open source repository bug software. But the end result needs to be compared with existing classifications or with others.

Nilesh Zaware et. Al also conducted online bug tracking research in 2016 using help software such as BugZilla, Trac, Mantis, BugTracker.Net, Gnats and Fossil. In addition, it also builds a database system using SQL Server, as well as Admin information. However, also the system built is compared with other systems that have been online such as Flyspray, Jtrac, Mantis, and phpBugTracker. The final results made by Nilesh Zaware are almost the same as Arvinder Kaur et. Al by making a system bug including collections on a large scale with the aim of being done automatically. In addition, it also runs on the Bugzilla software with other researchers. However, what was proposed by Arvinder Kaur was applied in C # with the data set taken from JIRA. These bug cases are analyzed and classified by various types such as security, memory and concurrency bugs, priority bugs, prediction of the number of bugs using machine learning. Furthermore, to measure its performance using a scale of precision, reliability and accuracy. Related to the classification also needs to be measured as proposed by Shruti Gujral et. Al by using a dictionary approach. However, technically it was initially traditionally done in testing the level of focus on bug classification. The classification of bugs is done using text mining techniques and Naïve Bayes Multinominal Classifier

III. RESEARCH METHODOLOGY

If we want to discuss about the bugs, there will always are gaps and even weaknesses. Beside that, Many and also several methods [21] and techniques that have been discovered and applied by many researchers, practitioners and academics. In addition to methods, problems and areas of research about bugs [22] are vary as first, classifying [23] different types of bugs, namely memory bugs, security bugs, and bug concurrency [24], [20], [25]. Second, priority bugs based on the status of bug reports, and then predict the severity of bug reports by using machine learning algorithms [6]. Although, in an effort to overcome the bug will be a complication due to the quality of relatively low bug reports that slow the handling of the bug itself. On other hand, bug problems not only can be applied with qualitative but also quantitative methods [26]. In This study, we would discussed were 4 type of features that fixed bugs and about requests for additional

information can significantly affect the time of bugs. The Cases of bugs have a negative impact and are even taken seriously. Therefore, The negative effects are critical bugs, big bugs, small bugs, and trivial bugs.

Currently researchers, developers, academics, and practitioners need to be aware and consider these bugs [27], [28] to be serious and get more attention [29] even if they need to be issued (Non-Reproducible Bugs), but this requires [30], [19] more profound techniques. Therefore, we will reduce the bugs that have high potential to low levels [31], [2]. The following Some the stages or phase will be conducted as followed .

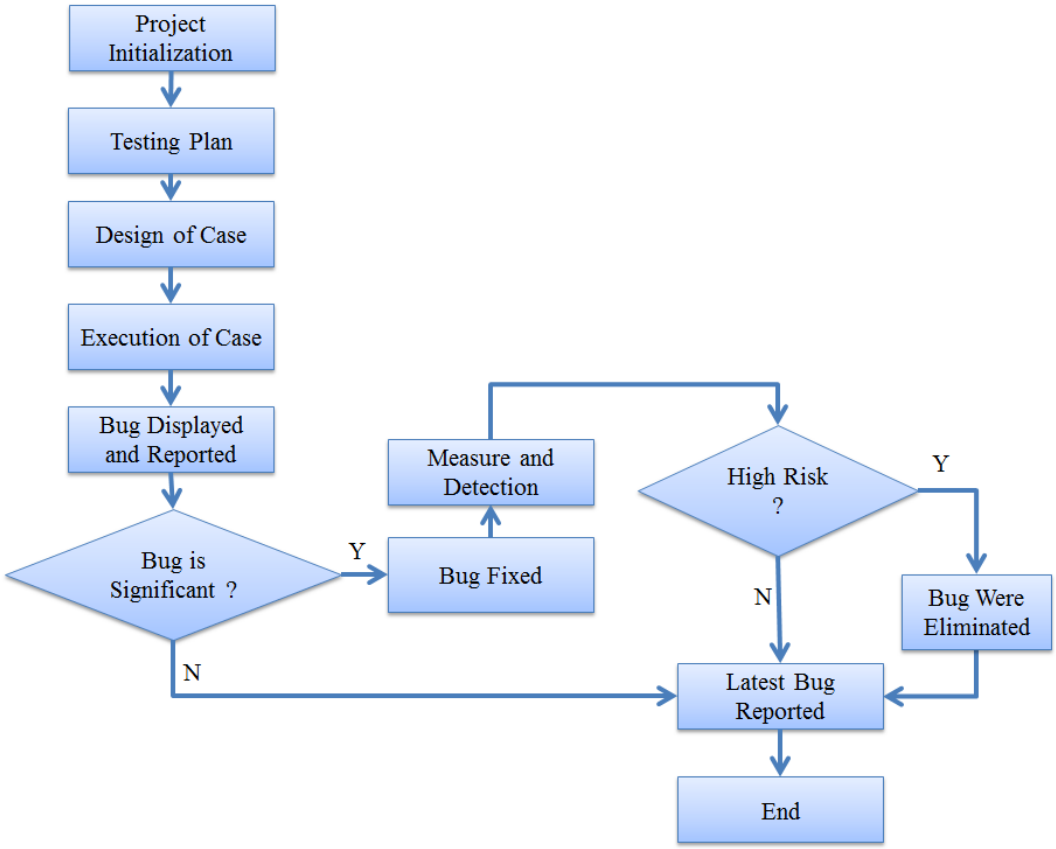

Figure 1. Flow of Process in Bug Simulation and Experiment

The process of eliminating bugs were considered as important, the main step by carrying out the project initialization stage followed by generating a trial plan. Next, a trial design had made by a series of test cases that had been prepared. First results, we would observation the bugs condition and are analyzed at the level of addition through measurement and detection. Measurements were conducted using fuzzy logic whether there is an item bug has a high category, while these bugs would be eliminated and will get the latest status about bug information.

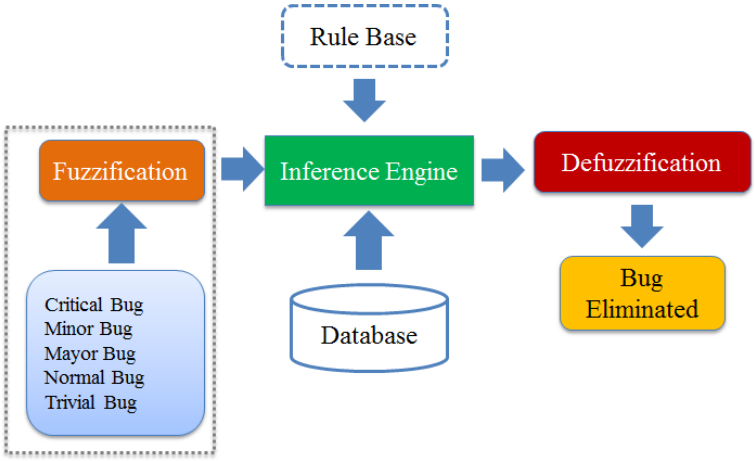

Figure 2. Fuzzy Logic Framework in Bug Simulation

Fuzzy logic as uncertainty method, was proposed by Lotfi Zadeh [27] [25], it is a method have been widely applied in any filed [32], and it has been adopted in the software testing case. One of which was adopted to make priority test cases [16] especially at the level of effectiveness. Umang Garg et. Al proposes fuzzy logic was applied to measure the estimation of trial capabilities, so that, it can prove to provide significant results. In This studies and related about the bugs, the fuzzy logic framework were proposed. The measurement phase starts with the input value through the verification phase with several indicators including critical bugs, minor bugs, major bugs, normal bugs, trivial bugs. The existing data set comes from the input value. Next, a logic inference arrangement will be made until the defuzification stage [33].

IV. Experiments Results

A. Design and Test Case

Initially, the bugs were considered trivial by some developers, system testers or some researchers. However, there are five bugs [34] that we identified based on the group: critical bugs, Minor Bugs, Major Bugs, Normal Bugs, Trivial Bugs. Critical bugs can cause programs to crash, data will be lost, even severe memory leaks [35]. Major Bugs are considered as bugs generated by large-scale functions. Whereas, trivial bugs were considered a type of bug that makes errors in the structure or spelling that are incorrect. Finally, normal and minor bugs as bugs that appear normally during the bug cycle. Our attention to each bug is primarily a major bug, trivial and critical bugs.

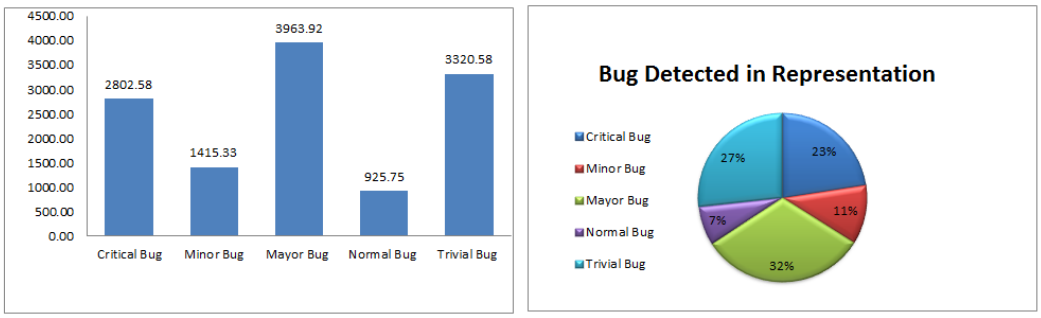

Figure 3. Reporting of bugs

Nearly 32% of major bugs rank at the same time as trivial bugs. This condition illustrates that the program we tested has sufficient potential.

B. Measurements and Bugs Prediction

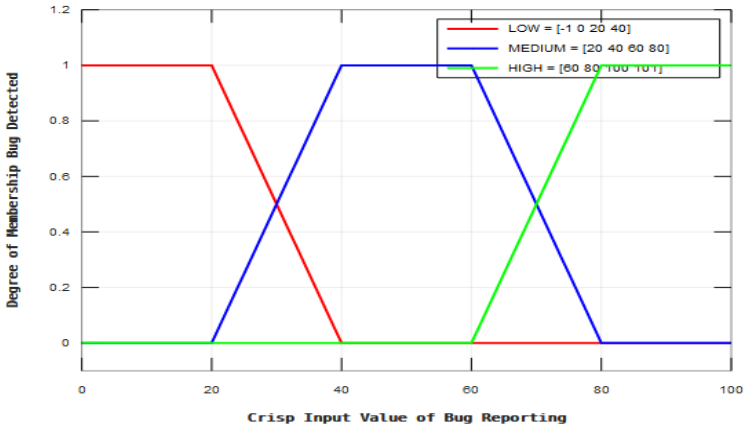

Figure 4. Membership Bug Detected

The bug that has been identified, its will be measured for severity through the fuzzy logic method. In this method, we determine the degree of membership with a range of values from 0 to 1 while the input values recommended in this logic start from 0 to 100. The identification results will be given indicators that are high, medium, and low. The range value between 0 - 40 is defined as a low indicator value and values 20 to 80 are indicated as middle values and values 60 to 100 are considered high class indicators. The data used during the trial process is done using the online octave version which is an open source application besides using matlab.

Table 1
Bug Summary in criteria

Karakteristik Kualitas	Bobot	Informasi Linguistik	μ
Critical Bug	0.94	High	1
Minor Bug	0.15	Low	0
Mayor Bug	0.70	High	1
Normal Bug	0.56	Medium	0
Trivial Bug	0.68	High	0

C. Bugs Eliminated and its effect

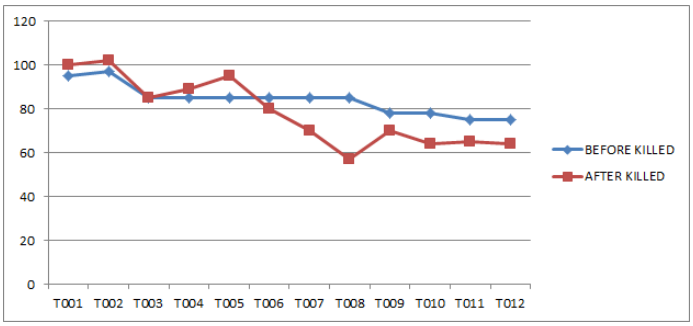

Figure 5. Main Status of the bug Reported

In the previous stage, we built a hypothesis that is

H0 :
The elimination process does not significantly affect the change in the number of bugs

Ha:
The elimination process has a significant effect on changing the number of bugs

As the next step, we test it with a paired t-test approach, where we provide 2 intermediate parameters before being eliminated and after being eliminated. Based on the picture above, it can be analyzed that the appearance of the initial bug has increased even though it is not significant. However, it applies also when the elimination process is applied, the bug decreases. Therefore, the process of elimination is very effective in reducing or eliminating bugs along with software testing.

In an effort to ensure this method has accuracy, we tested it in accuracy, precision and reliability. In this measurement TP is given notation as true positive, FP as False positive, FN as False Negative, and TN as True Negative.

$$precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The value of accuracy, precision and recall can be showed in the figure below:

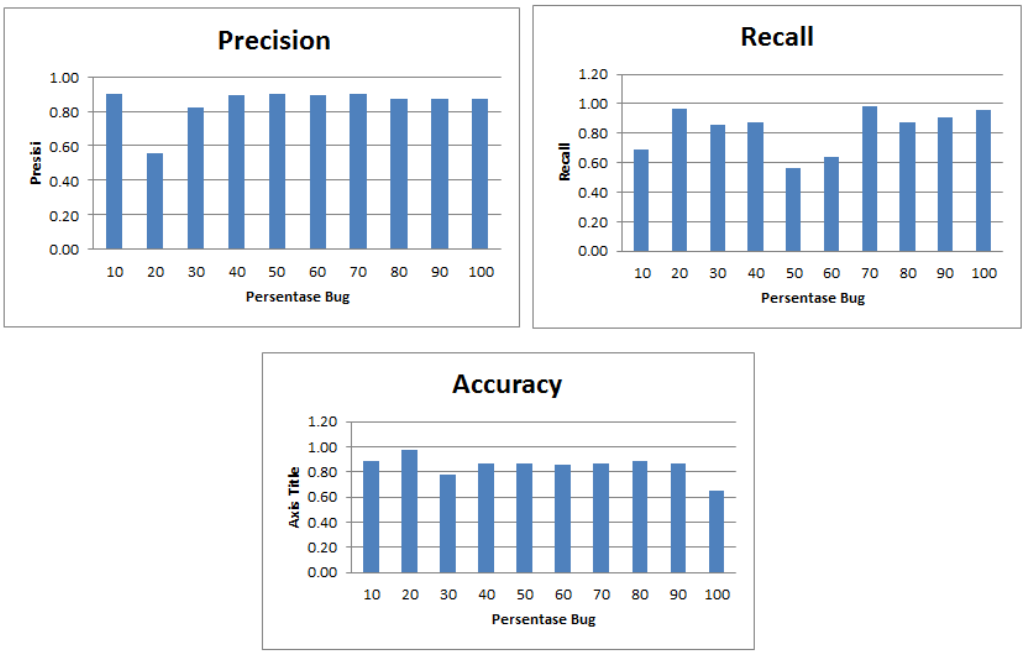

Figure 6. Precision, Accuracy, And Recall

The Bugs that have been dealt with can be seen that the precision level is almost 82%, which is supported by detection by using a recall that allows detection of new bugs. Therefore, the accuracy value can be trusted and supports the fuzzy method to be able to eliminate bugs that are considered to experience a significant increase.

V. CONCLUSION

Every developer wants to have software that has good quality and always attempted to avoid failure. The Bug problems are a major concern and provide a large business opportunity to overcome them. Besides that, Bugs cannot be lost and will continue to increase but there needs to be an effort to overcome them. In this approach, there are five criteria or aspects to be able to measure and reduce the bugs that occur. The fuzzy logic approach proved to be quite good in estimating the chances of bug occurring. Nearly 20% of bugs decrease when a bug is considered critical. Even though the bug is reduced but the risk of bugs persists. Therefore, it is better to reduce the scheduled bugs and even need a more effective strategy.

REFERENCES.

[1] D. M. Fernandes and J. H. Passoth, "Empirical Software Engineering: From Discipline to Interdiscipline," *The Journal of Systems & Software,* vol. 148, pp. 170-179, 2019.

[2] Anjali, D. Mohan and N. Sardana, "Visheshagya: Time Based Expertise Model for Bug Report Assignment," in *International Conference on Contemporary Computing (IC3)*, Noida, India, 2016.

[3] Z. K. Aghdam and B. Arasteh, "An E±cient Method to Generate Test Data for Software Structural Testing Using Arti¯cial Bee Colony Optimization Algorithm," *International Journal of Software Engineering and Knowledge Engineering,* vol. 27, no. 6, pp. 951-966, 2017.

[4] N. Zaware, P. Datir, M. Balwadkar and V. Wadje, "Online Bug Tracking System," *International Research Journal of Engineering and Technology (IRJET),* vol. 3, no. 10, pp. 597-598, 2016.

[5] Amandeep and S. Mittal , "A REVIEW ON BUG TRACKING SYSTEM USING NAÏVE BYES IN DATA MINING," *International Journal of Advance Research In Science And Engineering,* vol. 3, no. 9, 2014.

[6] K. K. Chaturvedi and V. B. Singh, "An Empirical Comparison of Machine Learning Techniques in Predicting the Bug Severity of Open and Closed Source Projects," *International Journal of Open Source Software and Processe,* vol. 4, no. 2, pp. 32-59, 2012.

[7] T. F. Bissyand, F. Thung, S. Wang, D. Lo, L. Jiang and L. Reveillere, "Empirical Evaluation of Bug Linking," in *European Conference on Software Maintenance and Reengineering*, Genova, Italy, 2013.

[8] L. Deng and J. Offutt, "Experimental Evaluation of Redundancy in Android Mutation Testing," *International Journal of Software Engineering and Knowledge Engineering,* vol. 28, no. 11, p. 1597–1618, 2018.

[9] F. Peng, C. Li, X. Song, W. Hu and G. Feng, "An eye tracking research on debugging strategies towards different types of bugs," in *Annual Computer Software and Applications Conference*, 2016.

[10] S. Fujimoto, H. Kojima and T. Tsuchiya, "Controlling Occurrence Frequencies of Parameter Values in Pair-Wise Testing," *International Journal of Software Engineering and Knowledge Engineering,* vol. 28, no. 7, pp. 983-1000, 2018.

[11] A. M. Alakeel, "Using Fuzzy Logic in Test Case Prioritization for Regression Testing Programs with Assertions," *The Scientific World Journal,* pp. 1-10, 2014.

[12] A. Tsuruda, Y. Manabe and M. Aritsugi, " Can We Detect Bug Report Duplication with Unfinished Bug Reports?," in *Asia-Pacific Software Engineering Conference*, New Delhi, India, 2015.

[13] D. Chen, B. Li, C. Zhou and X. Zhu, "Automatically Identifying Bug Entities and Relations for Bug Analysis," in *International Workshop on Intelligent Bug Fixing (IBF)*, Hangzhou, China, 2019.

[14] R. Ozakinci and A. Tarhan, "Early Software Defect Prediction: A Systematic Map and Review," *The Journal of Systems & Software,* vol. 144, pp. 216-239, 2018.

[15] M. S. Rahim, A. E. Chowdhury, D. Nandi and M. Rahman, "Issue Starvation in Software Development: A Case Study on the Redmine Issue Tracking System Dataset," *Journal of Telecommunication, Electronic and Computer Engineering,* vol. 9, no. 3, pp. 185-189, 2016.

[16] A. kaur and S. G. Jindal, "Bug Report collection system (BRCS)," in *International Conference on Cloud Computing, Data Science & Engineering - Confluence*, Noida, India, 2017.

[17] G. Tokdemir and N. E. Cagiltay, "Investigating the Relationship Between SLOC and Logical Database Measures to Improve the Early Estimation of Software Cost," *International Journal of Software Engineering and Knowledge Engineering,* vol. 29, no. 3, pp. 401-413, 2019.

[18] H. Yang, F. Chen and s. Aliyu, "Modern Software Cybernetics: New Trends," *The Journal of Systems & Software,* pp. 1-32, 2016.

[19] Z. A. Shaffiei , M. Mokhsin and S. R. Hamidi , "Change and Bug Tracking System: Anjung Penchala Sdn. Bhd.," *International Journal of Computer*

Applications, vol. 10, no. 3, p. 0975 – 8887, 2010.

[20] C. E. N. Gal-Chiş and B. Pârv, "MultiCoS - A Requirements Engineering Tool," *International Journal of Software Engineering and Knowledge Engineering,* vol. 28, no. 1, pp. 37-55, 2018.

[21] Kanaklata and S. Sharma, "Survey and Study of various Bug Tracking and Logging Toolkits," *International Journal of Computer Applications,* vol. 116, no. 12, p. 0975 – 8887, 2015.

[22] A. Goyal and N. Sardana, "Characterization Study of Developers in Non-Reproducible Bugs," in *Eleventh International Conference on Contemporary Computing (IC3)*, Noida, India, 2018.

[23] Y. Sharma, Shataksi, Palvika, A. Dagur and R. Chaturevedi, "Automated Bug Reporting System In Web Applications," in *2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, 2018.

[24] M. P. Fransisco, P. B. Perez and G. Robles, "Correlation between bug notifications, messages and participants in Debian's bug tracking system," in *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, Madrid, Spain, 2007.

[25] U. Garg and A. Singhal, "An Estimation of Software Testability using Fuzzy Logic," in *6th International Conference - Cloud System and Big Data Engineering (Confluence)*, 2016.

[26] T. Zhang and B. Lee, "A Bug Rule Based Technique with Feedback for Classifying Bug Reports," in *IEEE 11th International Conference on Computer and Information Technology*, Pavos, Cyprus, 2011.

[27] A. Tamrawi, T. T. Nguyen, J. Kofani and T. N. Nguyen, "Fuzzy Set-based Automatic Bug Triaging (NIER Track)," in *33rd International Conference on Software Engineering (ICSE)*, Honolulu, HI USA, 2011.

[28] S. Gujral, G. Sharma, S. Sharma and Diksha, "Classifying bug severity using dictionary based approach," in *International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, Noida, India, 2015.

[29] Jyoti and S. Hooda, "Optimizing Software Testing using fuzzy logic in Aspect oriented programming," *International Research Journal of Engineering and Technology (IRJET),* vol. 04, no. 04, pp. 3172-3175, 2017.

[30] T. Sultan, A. E. khedr and M. Sayed, "A Proposed Defect Tracking Model for Classifying the Inserted Defect Reports to Enhance Software Quality Control," *Acta Inform Med,* vol. 21, no. 2, pp. 103-108, 2013.

[31] L. K and J. P, "Bug Tracking System Analysis," *International Journal of Innovative Research in Computer and Communication Engineering,* vol. 3, no. 7, pp. 7038-7041, 2015.

[32] S. Kumar and P. Ranjan, "A Proposed Methodology for Phase Wise Software Testing Using Soft Computing," *International Journal of Applied Engineering Research,* vol. 12, no. 24, pp. 15855-15875, 2017.

[33] R. K. Saha, J. Lawall, S. Khursid and D. E. Perry, "Are These Bugs Really "Normal"?," in *Working Conference on Mining Software Repositories*, Florence, Italy, 2015.

[34] R. Karim, A. Ihara, X. Yang, H. Lida and K. Matsumoto, "Understanding Key Features of High-Impact Bug Reports," in *International Workshop on Empirical Software Engineering in Practice (IWESEP)*, Tokyo, Japan, 2017.

[35] M. S. Rahim, A. E. Chowdhury, D. Nandi and M. Rahman , "Issue Starvation in

Software Development: A Case Study on the Redmine Issue Tracking System Dataset," *Journal of Telecommunication, Electronic and Computer Engineering,* vol. 9, no. 3, pp. 185-189, 2016.

[36] G. Tokdemir and N. E. Cagilta, "Investigating the Relationship Between SLOC and Logical Database Measures to Improve the Early Estimation of Software Cost," *International Journal of Software Engineering and Knowledge Engineering,* vol. 29, no. 3, pp. 401-413, 2019.

[37] G. R. Santhanam, "Qualitative optimization in software engineering: A short survey," *The Journal of Systems and Software,* vol. 111, pp. 149-156, 2016.

[38] R. K. Saha, S. Khursid and D. E. Perry, "An Empirical Study of Long Lived Bugs," in *Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, Antwerp, Belgium, 2014.

[39] A. V. Rezende, L. Silva, A. Britto and R. Amaral, "Software Project Scheduling Problem in the Context of Search-Based Software Engineering: A Systematic Review," *The Journal of Systems & Software,* vol. 155, pp. 43-56, 2019.

[40] N. K. Nagwani, S. Verma and K. K. Mehta, " Generating taxonomic terms for software bug classification by utilizing topic models based on Latent Dirichlet Allocation," in *Eleventh International Conference on ICT and Knowledge Engineering*, Bangkok, Thailand, 2013.

[41] L. Merino, M. Ghafari, C. Anslow and O. Nierstrasz, "A Systematic Literature Review of Software Visualization Evaluation," *The Journal of Systems & Software,* vol. 144, pp. 165-180, 2018.

[42] Y. Liu, M. Li, Y. Wu and Z. Li, "A weighted fuzzy classification approach to identify and manipulate coincidental correct test cases for fault localization," *The Journal of Systems and Software,* vol. 151, pp. 20-37, 2019.

[43] H. Li, G. Gao, R. Chen, X. Ge and S. Guo, "The In°uence Ranking for Testers in Bug Tracking Systems," *International Journal of Software Engineering and Knowledge Engineering,* vol. 29, no. 1, pp. 93-113, 2019.

[44] K. Lavanya, P. Jennifer and T. Nadu, "Bug Tracking System Analysis," *International Journal of Innovative Research in Computer and Communication Engineering,* vol. 3, no. 7, pp. 1-4, 2015.

[45] T. R. Kumar and T. S. Rao, "Software Defects Prediction based on ANN and Fuzzy logic using Software Metrics," *International Journal of Applied Engineering Research,* vol. 12, no. 19, pp. 8509-8517, 2017.

[46] k. Kumar and Amandeep, "Bug Tracking System," *International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences,* vol. 2, no. 5, pp. 27-33, 2014.

[47] J. L. Davidson, N. Mohan and J. Carlos , "Coping with Duplicate Bug Reports in Free/Open Source Software Projects," in *IEEE Symposium on Visual Languages and Human-Centric Computing*, Pittsburgh, USA, 2011.

[48] K. K. Chaturvedi and V. B. Sngh, "An Empirical Comparison of Machine Learning Techniques in Predicting the Bug Severity of Open and Closed Source Projects," *International Journal of Open Source Software and Processes,* vol. 4, no. 2, pp. 32-59, 2012.

[49] Y. C. Cavalcanti and I. d. C. Machado, "A Systematic Literature Review of Software Visualization Evaluation," *The Journal of Systems & Software,* vol. 115, pp. 82-101, 2016.

[50] Amandeep and S. Mittal , "A REVIEW ON BUG TRACKING SYSTEM USING NAÏVE BYES IN DATA MINING," *International Journal of Advance Research In Science And Engineering,* vol. 3, no. 9, pp. 294-304, 2014.

[51] A. Patel , N. Pavasiya, C. Patil and . P. Sankpal , "Online Defect Tracking System (Under the guidance of Prof L.J Sankpal,Sinhgad Academy of Engineering KondhwaBk, Pune)," *International Journal of Scientific & Engineering Research,* vol. 4, no. 4, pp. 1861-1862, 2013.

[52] R. Malhotra and A. Chug, "Software Maintainability: Systematic Literature Review and Current Trends," *International Journal of Software Engineering and Knowledge Engineering,* vol. 26, no. 8, pp. 1221-1253, 2016.

[53] T. Zimmermann, R. Premraj, J. Sillito and S. Breu, "Improving Bug Tracking Systems," in *International Conference on Software Engineering*, 2014.

[54] K. Pandey, "A Bug Tracking Tool for Efficient Penetration Testing," *I.J. Education and Management Engineering,* vol. 3, no. 3, pp. 14-20, 2018.

[55] G. M. Puranik, "Design of Bug Tracking System," *nternational Journal of Innovative Research in Science, Engineering and Technology,* vol. 3, no. 7, pp. 14693-14696, 2014.

[56] K. Kumar and Amandeep, "Bug Tracking System," *International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences (IJIRMPS),* vol. 2, no. 5, pp. 27-33, 2014.

[57] H. Li, R. Chen, X. Ge and S. Guo, "The In°uence Ranking for Testers in Bug Tracking Systems," *International Journal of Software Engineering and Knowledge Engineering,* vol. 29, no. 1, pp. 93-113, 2019.