IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Volume-4, Issue-6 E-ISSN: 2347-2693

Auto-BugTriager for Software Development and Maintenance in Domain Specific Environment

K. Reshma Revathi[1*], S. Kirubakaran[2], R. Parimala[3], K.Sree Poornalinga[4], and K. Maheswari[5]

[1, 2, 3, 4] INFO Institute of Engineering
[5] SNS College of Technology, Coimbatore, Tamil Nadu, India

www.ijcseonline.org

Received: May/22/2016 Revised: May/30/2016 Accepted: Jun/14/2016 Published: Jun/30/ 2016

***Abstract*— The term bug triage refers to the process of assigning the suitable developer to fix the new incoming bugs which is considered to be one of the tedious process in software testing world than fixing a particular bug. The amount of new incoming bugs increases day-by-day, therefore manual bug triage becomes a tedious and imprecise for practical use thus increasing both the cost of development and time. In order to systematize the bug triage process, a wide number of data mining techniques can be used. Text Categorization is one of the effective method that helps in automating the system, where data reduction lies as one of the major difficulty identified in automating the bug triage process using text classification. Data reduction problem drops the exactness of automated bug triage prediction. This paper introduce a newly proposed framework entitled as Auto-BugTriager that aims to eliminate the data reduction problem better in range. The Auto-BugTriager framework is made up of three phase's viz. InfoZie, DataReduction and NBPredictor which process in an organized manner in order to predict the list of expert developer who is capable of fixing the particular bug in short time. As per the experimental and theoretical analysis, the proposed system provides 92% of accuracy in predicting the correct developer to fix a bug. Therefore, it is strongly believed that the industry adopting this framework will be improved with software productivity and quality.**

Keywords-Bug Triage Process, Manual Triaging, Data Mining Techniques, Text Classification, Data Reduction, Domain Specific Environment

I. INTRODUCTION

Software companies compacts with the flow of bugs in the whole thing of incoming or ongoing projects. For example, open source environment faces almost 300 bugs every day while the domain specific environment encounters nearly 30 to 100 bugs on daily basis. The proliferation of increase in arising bug solely depends on the size of the projects. This is much for programmers to handle by themselves as one may encounter numerous number of defects/bugs during build time. Also, the bug repository contains many bug reports of all kinds, mostly which are invalid or duplicate of another bug report. Therefore, each bug report in the repository must be validated for duplication and then needs to be triaged. In real time more than fixing a new bug, assigning that bug to an appropriate developer to fix is a time-consuming process. In traditional software development, triaging is done manually by an expert developer, i.e., a human triage. Due to the large number of daily bugs and the lack of expertise of all the bugs, manual bug triage is expensive both in time and cost [10]. In human triage whenever a new bug appears, an expert assigns the new bug to a developer, who will try to fix this bug. Most of the time the prediction of expert is not accurate. Manual triage is error-prone due to the large number of daily bugs and the lack of knowledge about all bugs by the developers. Human bug triage results in expensive time loss, high cost and low accuracy. Reassigning the new bug for different developers to fix lasts for months while fixing that bug takes only 2 to three days. Therefore, it is important to automate the bug triage process in every software companies for improving their production quality [11]. Research trend employs data mining concepts to deal with the software engineering problems. Many mining concepts such as text classification, fuzzy logic, text mining, extraction methods, etc., are being applied to automate the bug triage process.

Bug reports are free-form of data, which has two main challenges [10]. First challenge is the duplicate reports available in the bug repository. Mining large scale data will only results in low accuracy. The second challenge is the low quality of data i.e., the uninformative stop words in the bug report. Both these challenges are together are called as data reduction problem, which degrades the accuracy of bug triage. In this paper, a framework called Auto-BugTriager is proposed which aims to eliminate the problem of data reduction to greater extent in automatic bug triage process. The proposed framework consist of three phases namely InfoZie, DataReduction and NBPredictor which works together to predict the recommendation list of expert developer for fixing the bug [10]. In order to extend our idea for practical use, we propose to implement this framework in a domain specific environment. An experimental study is done implementing the Auto-BugTriager at the backend process of and Application system. The prediction of appropriate developer is measured to be almost 92.8%

accurate. The complete analysis report from the experimental study of the implemented system is provided at the end of this paper. As bug triage becoming more serious problem, organization face loss of both cost and time. Therefore, it is important to research over this problem for the improvement of software productivity and quality.

II. PROBLEM IDENTIFICATION

A. *Aim of the paper*

The objective of this paper is to study and analyze the data reduction problem in automating the process of bug triage. We propose to design a framework called Auto-BugTriager which aims to eliminate the problem of data reduction to greater extent in bug triaging.

- To improve the accuracy of bug triage.
- To extend my idea for real-time use.

B. *Issues*

Manual bug triage is expensive both in terms of cost and time. Therefore, automatic bug triage approaches are being proposed using data mining techniques such as text classification. The accuracy of these automatic approaches doesn't provide accuracy of more than 60% to 65%. One reason for the low accuracy is the problems of data reduction i.e. to reduce the bug dimension (Redundant data) and the word dimension (Data quality). Removing the duplicate bug reports completely from the bug repository reduces the accuracy of triaging due to the loss of valuable information's in the duplicate reports. Also, all noise data's are not removed, where data quality is not improved and leads to less accuracy. Up till now every organization use manual triaging and no industry has automated the process of bug triaging, which reduces the software productivity and quality. Also, the current researchers focus only on open source projects.

III. PROPOSED SYSTEM

The new Auto-BugTriager framework targets to eliminate the problem of data reduction in automating the bug triage process. The proposed framework be made up of three phases, viz. InfoZie, DataReduction and NBPredictor that works together in order to predict the list of top ten expert developer who are appropriate to fixing the bug.

A. *InfoZie*

InfoZie is the chief phase of AutoTriager that act as the validating tool. Duplicate bug reports often comprises of valuable extra information. So the duplicate reports are validated for the availability of extra information using InfoZie. Here, the master report and duplicate report are compared using a Diff based algorithm [10]. Diff algorithm is a UNIX based technique that compares the two files of same version for extra information. The InfoZie tool is based on the concept of diff algorithm. If duplicate report contains extra valuable information, it is merged with the original master report otherwise ignored. InfoZie improves the bug dimension as well as the accuracy of bug triage.

B. *DataReduction*

DataReduction involves in performing the following two specific processes.

- **Pre-processing:** Here, text classification technique is used to convert the summarized bug reports into text matrix where each row indicates one bug report, and each column indicates one word [10].
- **Noise Reduction:** This phase makes use of feature extraction method to reduce the word dimension i.e., removes the uninformative stop words from the bug reports [10].

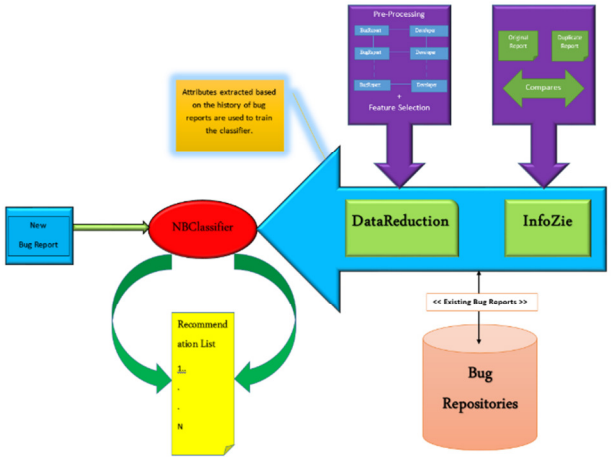

Figure 1: Auto-BugTriager Framework

C. *NBClassifier*

NBClassifier is the final phase of the Auto-BugTriager framework which predicts the list of expert developer who are appropriate in fixing the assigned bug. NBPredictor is created using Naïve Bayes classifier. [1, 10] The classifier is trained with the history attributes extracted from existing bug reports. The attributes of both in terms of technical aspect and developer details were extracted. The new bug report act as the test set which fed as input to the Auto-BugTriager.

We propose to extend this idea for real-time use by implementing the Auto-BugTriager framework at backend process of an application system. We implement the system in java using Eclipse platform to as it provide a platform independent environment to run the application system. This system helps the software developing team in managing and

organizing the bug reports by adding extra supporting features [10].

IV. METHODOLOGY

Data mining concepts can play a vital role in automating the bug triaging process in most efficient way. InfoZie, Pre-Processing, Feature selection and NB predictors are the key methodologies used in the proposed framework called Auto-BugTriager. A clear and deep theoretical study of each methodology has been provided below.

A. Data Pre-processing

The first step in processing the bug reports using the Auto-BugTriager framework is the data pre-processing. As the real world data is incomplete and inconsistent, mining the raw data is tedious and error prone process, which is one of the reasons for the decrease in accuracy of the automatic triaging. Therefore, transforming the raw data into an understandable format will make the mining process easy and thus accuracy can be achieved. The bug reports have a standardised format which is divided into reasonable categories. Therefore, the proposed framework uses relational database format which provides a convenient way for transforming each bug reports into number of rows and columns.

B. Feature Selection

One of the data reduction problems in automating the bug triage process is the word dimension i.e. the data quality. By nature, bug reports are in natural language, thus containing noisy information. The presence of noisy data decreases the quality of data and results in inaccuracy of triaging. Therefore, removing such noise information from the bug data is important process for increasing the accuracy of triaging process. Feature selection is the appropriate technique that provides attribute selection is the best way to improve the quality of data thus making the triaging process accurate. The proposed framework applies the feature selection aka attribute selection in two phases of the proposed framework. At first, the attribute selection is performed based on the selection of stop word. Stop words are the words that reside in the bug reports as noisy data that makes the mining process difficult and inaccurate. Therefore, removing such stop words is inevitability. This will improve the data quality and ease the mining process. Thus accuracy of triaging is improved. Also, another set of attribute selection is done based on the technical terms which are used for training classifier for prediction algorithm. This selected attributes are used in the third phase called NBClassifier. The feature selection technique helps the Auto-BugTriager to improve quality of bug data and remove all the noise information from the bug reports. One of the data reduction problems is likely to be eliminated with proper usage of feature selection technique, thus improving the accuracy of bug triage to greater extent.

C. InfoZie using Diff Algorithm

One of the main problem in automating the bug triage process using data mining technique is the redundancy of data due to the duplication of bug reports that resides in the bug repositories. The removal of duplicate reports is important, but all the duplicate reports in the bug repositories are not always a copy of original. Most of the times these duplicate reports are re-submitted intentionally by the programmers with some corrections or additional information. Therefore, there is 90% of chances for the presence of extra valuable information in the duplicate reports. Removing such valuable information will decrease the accuracy of bug triage. Therefore, it is significant to validate the duplicate reports obtainable in the centralized bug repository. The validation is performed by using a specific tool that as in [2] called InfoZilla which validates the duplicate reports through the extracted structural information from the bug reports. In proposed system, a tool called InfoZie based on diff algorithm (diff is a data comparison tool used to show the changes between two versions of the same file) that compares the technical elements extracted from both duplicate and master bug reports. According to InfoZie, if duplicate report contains spare data, it is then merged with the master bug report otherwise ignored. This improves the bug dimension and also the accuracy of bug triage buy handling the duplicate bug reports [10]. The modified diff algorithm for the InfoZie tool is given below, this algorithm works as following assumptions. Consider two data sets named NewBugReports and BugRepository where the former holds the new bug reports submitted and the latter holds all the existing bug reports along with its developer history.

```
For each Bugdata in NewBugrepository
     CompareNew data with each Master data
in BugRepository
          If there is a match sequence
               If (mirrored data found)
                    Merge the result
               Else//Good best match found
                    ShowReport,
                    ShowOptions(Merge,
                                Skip)
         Else
              Skip//No matches found
        End If
End For
```

Algorithm 1. Generic algorithm for Diff based InfoZie Alg.

The InfoZie phase in AutoTriager framework is based on the above modified diff algorithm. This will completely remove

the duplicate reports from the bug repositories without any loss of valuable information thus decreasing the bug dimension i.e. the redundant data and increasing the quality of bug data. This is the heart of Auto-BugTriager that helps in increasing both the quality of data and accuracy of triaging. The analysis report of the InfoZie tool has been reported clearly in the latter section.

D. Naive Bayes Classifier

The final phase of the Auto-BugTriager framework uses the Naive Bayes classifier for predicting the recommendation list. Naive Bayes classifier is easy to build and particularly useful for very large data sets. Therefore, this is an appropriate classifier for triaging bug reports which increases in rapid amount from day to day in corporate world. NBClassifier taht predicts the recommendation list for the new bug report i.e. the NBPredictor, which is created using Naïve Bayes classifier. The classifier is trained with the history attributes extracted from existing bug reports. The attributes of both in terms of technical aspect and developer details. Here the new bug report is the test set which is input to the AutoTriager.

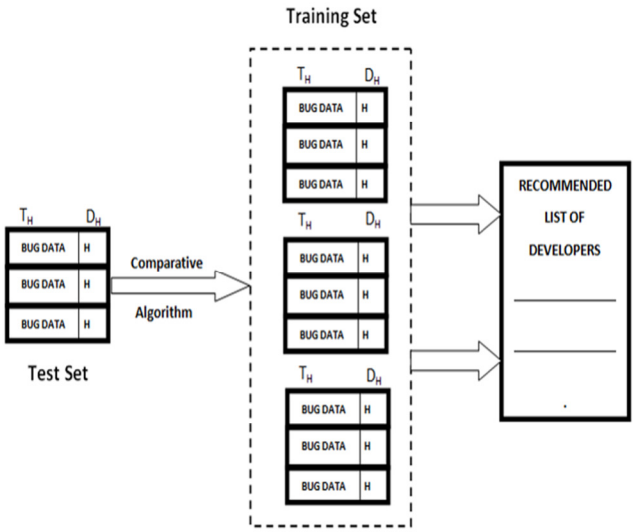

Figure 2: Background Task of NBClassifier

Naive Bayes is simple and also identified to overtake the uniformly refined taxonomy approaches. Figure 2 shows the working analysis of NBClassifier which is trained with the history attributes extracted from existing bug reports. T_H and D_H are the two attributes namely Technical and Developer History. Here the new bug report is the test set which is input to the Auto-BugTriager. The output is the list of recommended developers who can fix the new similar bug.

V. RESULT AND DISCUSSIONS

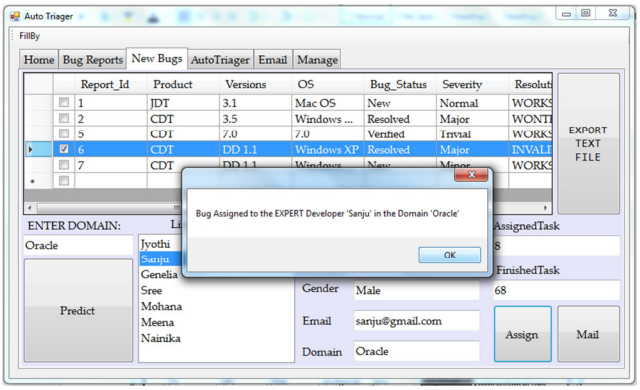

Figure 3. Auto-BugTriager Implemented For Practical Use

Experimental study for the Auto-BugTriager has been performed thoroughly. The monitored analysis report from the implemented system has been provided in detail here. The experiment is done for different set of datasets for analysing the quality, and improvement of accuracy. The performance analysis of the implemented Auto-BugTriager has also been given in detail for reference. The data reduction problem in automating the bug triage process has been eliminated almost. Figure 4 is the graph which shows the detection rate of redundant information examined for different sets of sample datasets. Nearly 96.7 percent of the duplicate reports has been successfully detected without any loss of valuable information, with the help of InfoZie tool.

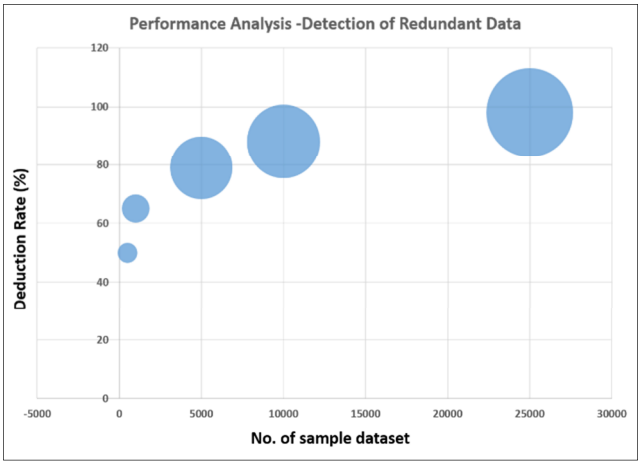

Figure 4. Detection of data redundancy experimented for different sets of sample bug dataset.

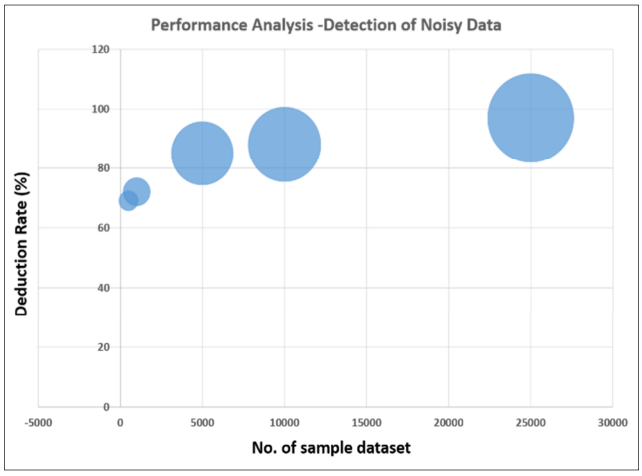

Figure 5. Detection of noise data experimented for different sets of sample bug dataset.

Similarly, the detection of noise data has also been experimented and the analysis report is shown in the figure 5. The experiment is done for different number of sample bug reports starting from 5000 to 25000 bug reports. The bug report analysis, both before and after the removal of data reduction problem is compared in Figure 6. The comparison clearly explains the increase in data quality after the Auto-Bug Triage process. The performance rate reaches almost 98.7 percentage of data quality. When data quality is high, the triaging process will be accurate with its result. Figure 3 is the experimental analysis, implemented to analyse the working of the prosed Auto-BugTriager for practical use. The analysis report clearly shows that proposed framework called Auto-BugTriager has reached 96.3 percent of accuracy.

Figure 6. Performance analysis for data quality before and after data reduction

VI. CONCLUSION AND FUTURE WORK

As explained previously, data reduction is one of the key problem in applying text categorization to systematize the bug triage process, which shrinks the exactness of bug triaging. This paper introduces a newly proposed framework called Auto-BugTriager which targets to eliminate the problem of data reduction in greater amount. Automating the bug triaging process using Auto-BugTriager that consist of three phases viz., InfoZie, DataReduction and NBPredictor that works in a combined form to predict the list of graded expert developer for who can fix the newly assigned bug within a short timespan without letting the reassigning process to occur. The proposed framework is applicable for all kinds of software project environment that faces bulky amount of bugs on daily basis. It has been strongly believed that the proposed system could help software organizations in improving mutually the quality and accuracy of bug triage in testing environment.

A complete theoretical and Experimental analysis of the proposed framework is reported in this paper. As per the report, the quality of data is improved rapidly. The data reduction problem has been almost eliminated in automating the process using text classification and also results in improved accuracy of triaging process. In our future work, we propose to create a new infrastructure for achieving complete accuracy for both domain specific and open source bug reports. We also tend to implement this model real time industrial data.

REFERENCE

[1] JifengXuan, He Jiang, Yan Hu, ZhileiRen, WeiqinZou, ZhongxuanLuo, and Xindong Wu,"Towards Effective Bug Triage with Software Data Reduction Techniques" IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 1, January 2015

[2] Nicolas Bettenburg and Rahul Premraj, "Extracting Structural Information from Bug Reports," May 2008,

[3] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Duplicate bug reports considered harmful ... really?" in ICSM08: Proceedings of IEEE International Conference on Software Maintenance, 2008, pp. 337–345.

[4] V. Bolon-Canedo, N. Sanchez-Maro~no, and A. Alonso-Betanzos,"A review of feature selection methods on synthetic data," Knowl. Inform. Syst., vol. 34, no. 3, pp. 483–519, 2013.

[5] D. Cubranic and G. C. Murphy, "Automatic bug triage using text categorization," in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng., Jun. 2004, pp. 92–97.

[6] T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction," in Proc. 22nd IEEE Int. Conf. Tools Artif. Intell., Oct. 2010, pp. 137–144.

[7] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," ACM Trans. Soft. Eng. Methodol., vol. 20, no. 3, article 10, Aug. 2011.

[8] S. Kim, K. Pan, E. J. Whitehead, Jr., "Memories of bug fixes," in Proc. ACM SIGSOFT Int. Symp. Found. Softw. Eng., 2006, pp. 35–45.

[9] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: Improving cooperation between developers and users," in Proc. ACM Conf. Comput. Supported Cooperative Work, Feb. 2010, pp. 301–310.

[10] K. Reshma Revathi and Dr. S. Kirubakaran, "A Survey on Automatic Bug Triage Using Data Mining Concepts", International Journal of Science and Research (IJSR), ijsr.net, Volume 5 Issue 3, March 2016, 184 – 186.

[11] K. Sree Poornalinga, P. Rajkumar, "Survey on Continuous Integration, Deployment and Delivery in Agile and DevOps Practices", International Journal of Computer Sciences and Engineering, Volume-04, Issue-04, Page No (213-216), Apr - 2016, E-ISSN: 2347-2693

Authors Profile

Ms. K. Reshma Revathi pursed Bachelor of Science from Shri Nehru Maha Vidyalaya College of Arts and Science affiliated to Bharathiar University, India in 2007 to 2010 and Master of Computer Application from Info Institute of Engineering affiliated to Anna University in year 2010 to 2013. She is currently pursuing Master of Engineering in Computer Science and Engineering at Info Institute of Engineering affiliated to Anna University from 2014 to present. Coimbatore, Tamil Nadu, India. Her main research work focuses on Data Mining, Computer Networks and Software Engineering.

