AUTOMATIC BUG TRIAGING USING DATA REDUCTION TECHNIQUES FOR DYNAMIC BUG REPORTS

Pooja Chavan[1], Ashwini Gilbile[2], Rajat Deshpande[3], Devendra Bakare[4]

[1] *Student, Department of Computer Engineering, Dr. D.Y Patil College of Engineering, Akurdi, Maharashtra, India*
[2] *Student, Department of Computer Engineering, Dr. D.Y Patil College of Engineering, Akurdi, Maharashtra, India*
[3] *Student, Department of Computer Engineering, Dr. D.Y Patil College of Engineering, Akurdi, Maharashtra, India*
[4] *Student, Department of Computer Engineering, Dr. D.Y Patil College of Engineering, Akurdi, Maharashtra, India*

ABSTRACT

Triaging basically implies sorting any entity into groups. Triaging of bugs is manual, tedious and time consuming task. Mostly, majority of the developers are specialized in specific tasks. For example, some developers have expertise in java, some in GUI, few in python and so on. Therefore, triaging of bugs is extremely important. Assigning a bug to relevant developer would not only save time but would help the developers process bugs according to their requirements. It is very difficult for a triager to assign the bug to relevant developer without knowing to which class the bug belongs. In this paper, various bug triaging techniques are discussed for efficient bug triage.

Keywords: - *Bug Triage, Feature selection, Instance selection, Mining Software repository, Developer Recommendation.*

1. INTRODUCTION

Usually during project development, bugs are maintained and resolved using bug repository. A bug repository is a database of bug reports which needs to be solved. Today, in most of the cases the team lead of the project has to spend hours or a almost a day to decide the assignment of the bugs to relevant developer for resolution, taking preferences in consideration. This increases the overall workload on an individual. Assignment of a bug report to respective developer for solving a bug is a time consuming as well tedious job. The project Manager should have good knowledge about the expertise of the developer and also his preferences in fixing bugs, his current workload, the priority and the task assignment. It is very expensive task for assigning and resolving the bugs. Automatic bug triage, by applying text classification techniques, reduces the cost as well as the time of manual work. The technique proposed in paper combines instance selection with feature selection for reduction of data scale on the bug dimension as well as the word dimension. To determine the order of applying instance selection and feature selection, it extracts attributes from dynamic bug data sets and builds a predictive model for a new bug data set. The results show that the data reduction can effectively reduce the data scale and improve the accuracy of bug triage. The proposed system Forms a reduced and high-quality bug data in software development and maintenance by providing an approach to leveraging techniques on data processing. (Bug Life cycle) is the journey of a defect from its identification to its closure. The Life Cycle varies from organization to organization and is governed by the software testing process the organization or project follows and/or the Defect tracking tool being used.

2. LITERATURE SURVEY

The following section gives an overview of various research papers based on Bug Triaging:

A Bug You Like: A Framework for Automated Assignment of Bugs:

Olga Baysal et al proposed a set of heuristics in paper [1] which focused on higher accuracy in the assignment of bugs to the developers which uses a vector space model. The major use of this model is for selection purpose. The bugs on the arrival basis are recommended to relevant experts for resolving them based on expertise, current workload and preferences or preference elicitation comes under the field of artificial intelligence. It deals with acquisition of the preference of users and making decisions by using them on users' behalf. Preference elicitation is the trending topic of discussion among various research communities. It is a topic which has to be further explored for entire study. The main goal of this system is to explore how preference elicitation techniques can be applied to the field of software engineering and what challenges may arise during practical execution of PE techniques. The proposed framework selected the developer on the basis of bug history of that particular developer. The bugs resolved by the developer beforehand were taken into consideration and assigned accordingly. The only drawback of this system is it can not provide answers on questions like what and how to elicit preferences from the users in a specific domain. This is a major challenge.

A Dynamic Approach to Software Bug Estimation:

In paper [2], Hemant Joshi et al proposed a system that used already recorded bugs in the Eclipse project to predict the occurrences of the bugs that may arrive in future. Major problems of global software development and resource allocation in product development is taken into consideration. There can be a location difference between various entities involved in software development, like the developer and the tester. Hence it becomes a tedious task to conduct meetings for bug triage and its relevant assignment to the developer. Hence a record is maintained to facilitate the same. The record contains the history of occurrences of the bugs, and their subsequent resolution. The system makes use of a local and global recently weighting technique for the design of a bug prediction algorithm. This algorithm is tuned to work on a monthly basis. At the start of every month, bug reports and patterns are generated. In experimental results, patterns are obtained which help in project management to determine the need of resources in near future. The bug count of the current month depends on the bug count of the previous month. The experimental results of this system show some extremely undesired results, which made the system a bit less promising. Dynamically arriving new bugs could not be assigned to developers, as there was the use of pre-stored open source bug repository.

Who should fix this bug:

In paper [3], the authors have taken into consideration the problem of large datasets. It naturally is very difficult to work on the large datasets as compared to reduced size datasets. The main aim of this research is to employ various data reduction techniques and reduce the scale of data. The main techniques that are used are Instant selection and feature selection. For the application of instance selection and feature selection, there is an extraction of attributes from predefined bug data sets and building of a predictive model for a new bug data set. The experimental results depict that the data reduction can show efficiency in reducing the data scale and improvising in terms of accuracy of bug triage. The proposed work provides an approach to get a high quality and reduced bug dataset. However, the problem of bug assignment is slightly overlooked, and the bugs existing is the predefined repository are the ones only taken into consideration.

Automated Duplicate Detection for Bug Tracking Systems:

The paper [4] proposed a system that checks the duplicity in the bug reports. The authors have instantiated a model that automatically indicates whether an arriving bug report is original or duplicate of an already existing report. It

saves the developers time and efforts. To predict and avoid bug duplication, there is a proposed use of rudimentary methods such as textual semantics, clustering in graphs and surface features. The only drawback of the proposed system was its inability to prove effective, as it could filter out only 8 percent of bugs. Input of dynamic bugs, not present in the pre-existing repository of bugs is not taken into consideration.

Mining Bug Classifier and Debug Strategy Association Rules for Web-Based Applications:

The paper [5] proposed a technique by making use of data mining techniques to automatically classify the bugs of web based applications by predicting their bug type. Now days, Web based applications are at the peak of usage. It is very important to perform bug mining on these applications. The authors have proposed a data mining technique to obtain the bug classifier which will help in facilitating bug mining and developing debug strategy rules. Feature

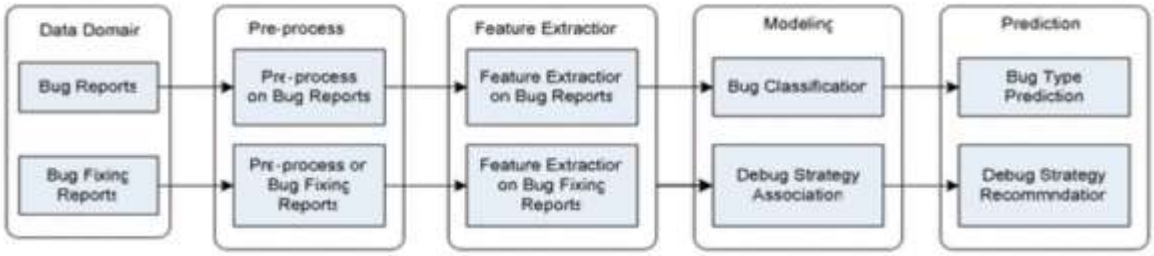

Fig. 1. Data Mining Process of Bug Classification and Debug Strategy Association Rules

Extraction methodology is implemented with the help of Chi-Square algorithm. For training dataset, the use of SVM (Support Vector Machine) is done. The debug strategy will help to find the relationship between bug types and bug resolution solutions. The debug strategy introduces us with the erroneous part of the code. Once the errors are discovered, then it is an easy task for the developers to resolve them. The drawback is, bugs from the Mozilla repository are used. Newly occurred bugs cannot be given as input for their respective mapping.

Software Escalation Prediction with Data Mining:

Tillman Bruckhaus et al have proposed a system for Escalation Prediction (EP) in paper [6]. Escalation is basically an error or defect, or more precisely a bug that affects the system and downgrades the core operation. These escalations are caused by the customer using the software. Such an escalation should be solved before there is a heavy damage to the user. Solving an escalation externally can cost a lot of effort and money. Hence there is a proposal of the EP model. The objective Fig 2: Escalation Prediction (EP) solution architecture of Escalation Prediction (EP) is to avoid such escalations from known product defects using data mining technology. This model can predict the risk of an escalation well before the occurrence and can also be fixed, without any significant damage. Though this paper [5] proposes a system which can predict the escalations, the practical effectiveness when examined is much lesser.

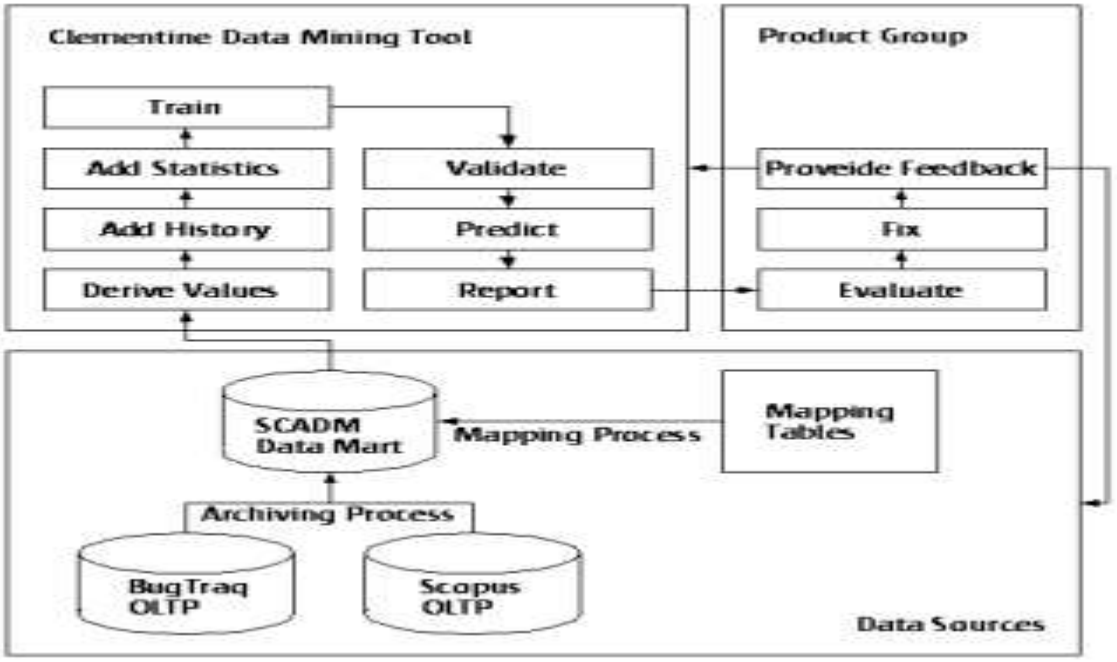

Fig 2: Escalation Prediction (EP) solution architecture

Improving training set reduction for bug triage:

In paper [7], they introduced a graph model based on Markov chains, which generally captures a bug tossing history data. This model reveals developer networks which can be used to discover the team structures and then find suitable experts for a new job. Next it will help to better assign developers to the bug reports. The experiments were done with 445,000 bug reports, the model reduced tossing events, by up to 72 percent. It also increased the prediction accuracy up to 23 percentage points compared to the traditional bug triaging techniques. This model captures the tossing probabilities between developers from the tossing history which is available in bug tracking systems Technique to Combine Feature Selection with Instance Selection for Effective Bug Triage In this paper bug reports are collected from Eclipse and Mozilla and evaluated for bug data reduction on bug repositories of these two large open source projects. For each bug report, we download web pages from bug repositories and extract the details of bug reports for experiments. Only fixed and duplicate bug reports were chosen for triaging. Moreover, in bug repositories, several developers have only fixed very few bugs. Such inactive developers may not provide sufficient information for predicting correct developers. In their work, they removed the developers, who had fixed less than 10 bugs. Their work removed noisy duplicate words in a data set with the help of feature selection but instance selection algorithm decreased the accuracy by removing uninformative bug reports.

Bug Triaging : Profile Oriented Developer Recommendation:

In paper [8] term frequency mining technique is used to predict the most suitable developer for the new bug report depending on information available from historical bug reports. The terms (or topics) extracted from the previously fixed bug reports are used for creation of profiles of developers. The frequency of each term corresponding to each topic indicates the expertise factor of each developer relating to that topic. Paper [5] proposed a Domain Mapping Matrix (DMM) based developer recommendation approach for predicting the best suited developers list who could resolve the newly reported bugs. Unlike other approaches, our approach does not find a matching with historical bug

reports and recommend the developers who fixed historical bug report; rather, it utilizes the expertise profile of developers maintained in DMM. This profile can be easily updated with time. But it is not known that whether accuracy of approach is more dependent on the terms extracted from bug reports or source code.

An Empirical study of real bug fixes:

The authors of paper [9] proposed a system called BUGSTAT, which was used for extraction, analysis and resolutions of various bugs found in the lifetime of a system execution. Various Java projects, including six major ones, were taken into consideration. Over 9000 bug reports were considered for the experimental results of the BUGSTAT tool. There was a major difference found between manual and automatic bug triage. The differences were further classified into two major parts of automatic bug repair, which are fault localization and fault code fixation. In addition to BUGSTAT, the authors propose a search space that can extract bugs from non-source files and provide further fixation. Although this research shows promising and effective results, its real-world application remains a big question, as bugs from only major Java projects have been taken into consideration. Limitation of bug fixation makes this project less effective as compared to the proposed system.

3. PROPOSED SYSTEM

The system proposed in this paper basically implements text classification techniques like Feature selection and Instant selection. In order to reduce high dimensionality of data various data reduction techniques are used. Depending upon the type of bug various departments need to be created. For example, to classify bugs of java, java department needs to be created for python related bugs python department needs to be created and so on. Coupling of data reduction algorithms along with classifier like SVM or nave Bayes enables faster mapping of bug reports to relevant developers.

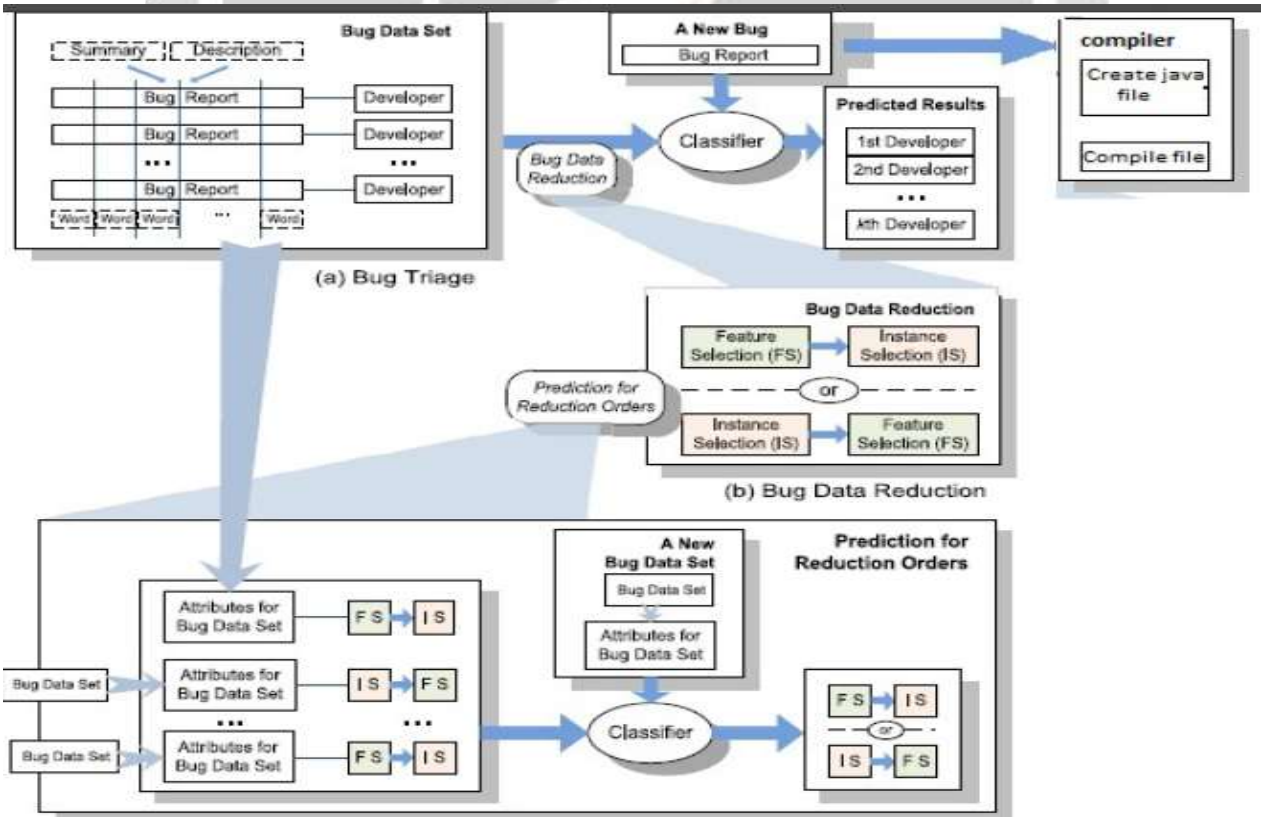

Fig 3: System Architecture

4. RESULT AND FUTURE SCOPE

The successful implementation of the proposed system enables faster mapping of the bugs to the relevant developer. When the bugs are submitted to the system, the system classifies the bugs according to technology been used like java, PHP, python etc. The system then assigns it to appropriate developer through the use of text classification and data reduction techniques. In time, the developers could learn how the system assigns bug fixing tasks and try to manipulate task assignment. Thus, we should clinch that assignment of bugs is a fair and manipulation-free process. The scope of proposed work is assigning developer based on priority levels to the new bug report and assigning proper ranking to the predicted list of developers. The Instance selection algorithm can remove uninformative bug reports whereas keyword selection can remove uninformative words. Keyword selection improves the accuracy of bug triage.

5. CONCLUSION

Triaging and resolving a software bug is an important step in software industries. It saves labor cost as well time. In this paper, a method is proposed that combines the feature selection algorithm (FS) with instance selection algorithm (IS), which trims down the scale of bug data sets. The proposed system performance is verified using dynamic bug data set. It improves the data quality. Future work includes preparation of a high-quality bug data set. The system automatically assigns a bug to the respective developer for fixation. Our work provides a reduced and high-quality bug data in software development and maintenance by providing an approach to leveraging techniques on data processing.

6. REFERENCES

[1] O. B. Michael and G. C. Robin, ”A Bug You Like: A Framework for Automated Assignment of Bugs.,” IEEE 17th international conference, 2009.

[2] C. Zhang, H. Joshi, S. Ramaswamy and C. Bayrak, ”A Dynamic Approach to Software Bug Estimation,” in SpringerLink, 2008

[3] J. Anvik, L. Hiew, and G. C. Murphy, Who should fix this bug? in Proc.28th Int. Conf. Softw. Eng., May 2006, pp. 361370.

[4] N. Jalbert and W. Weimer, Automated Duplicate Detection for Bug Tracking Systems, in IEEE computer society, 2008.

[5] L. Yu, C. Kong, L. Xu, J. Zhao and H. Zhang, ”Mining Bug Classifier and Debug Strategy Association Rules for Web-Based Applications,” in 08 Proceedings of the 4th international conference on Advanced Data Mining and Applications , 2008.

[6] Tillman Bruckhaus, Charles X. Ling, Nazim H. Madhavji and Shengli Sheng,”Software Escalation Prediction with Data Mining”, Fifth IEEE International IEEE Conference, 30 Nov. 2005

[7] G. Jeong, S. Kim, and T. Zimmermann, Improving training set reduction for bug triage, in Proc. 35th bug triage with tossing graphs, in Proc. Joint Meeting 12th Eur. Softw. Eng. Conf. 17th ACM SIGSOFT Symp. Found. Softw. Eng., Aug. 2009, pp. 111120.

[8] Anjali Sandeep Kumar Singh, Bug Triaging: Profile Oriented Developer Recommendation, (IJIRAE) ISSN: 2349-2163 Volume 2 Issue 1 (January 2015).

[9] Hao Zhong and Zhendong Su An Empirical study of real bug fixes, ICSE ’15 Proceedings of the 37th International Conference on Software Engineering - Volume 1 Pages 913-923