ISSN:0975-9646
Ashwini Jadhav et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (5) , 2015, 4611-4612

A Survey on Software Data Reduction Techniques for Effective Bug Triage

Ashwini Jadhav[1], Komal Jadhav[2], Anuja Bhalerao[3], Amol Kharade[4]

[1,2,3,4] *JSPM's Imperial College of Engineering, Wagholi, Pune, India*

***Abstract*— Most of the software companies needs to deal with large number of software bugs every day. Software bugs are unavoidable and fixing software bugs is an expensive task. The goal of effective bug triaging software is to assign potentially experienced developers to new-coming bug reports. To reduce time and cost of bug triaging, an automatic approach is proposed in this paper that predicts a developer with relevant experience to solve or fix the new coming bug report. In this paper, the five term selection methods on the accuracy of bug assignment are used. In addition, the load between developers based on their experience is re-balanced. The proposed system is built with intention to suggest or recommend the bug and not to automatically assign it. This allows a window to handle real time crisis that come up during project development lifecycle.**

***Keywords*— Mining software repositories, application of data pre-processing, data management in bug repositories, bug data reduction, feature selection, instance selection, bug triage.**

I. INTRODUCTION

Many software companies spend most of the money in fixing the bugs. Large software projects have bug repository that collects all the information related to bugs. In bug repository, each software bug has a bug report. The bug report consists of textual information regarding the bug and updates related to status of bug fixing.

Once a bug report is formed, a human triager assigns this bug to a developer, who will try to fix this bug. This developer is recorded in an item assigned-to. The assigned-to will change to another developer if the previously assigned developer cannot fix this bug. The process of assigning a correct developer for fixing the bug is called bug triage. Bug triage is one of the most time consuming step in handling of bugs in software projects.

Manual bug triage by a human triager is time consuming and error-prone since the number of daily bugs is large and lack of knowledge in developers about all bugs. Because of all these things, bug triage results in expensive time loss, high cost and low accuracy.

The information stored in bug reports has two main challenges. Firstly the large scale data and secondly low quality of data. Due to large number of daily reported bugs, the number of bug reports is scaling up in the repository. Noisy and redundant bugs are degrading the quality of bug reports.

In this paper an effective bug triage system is proposed which will reduce the bug data to save the labor cost of developers. It also aims to build a high quality set of bug data by removing the redundant and non-informative bug reports.

II. LITERATURE SURVEY

In [1] they mention that Bug triaging is an error-prone, tedious and time consuming task. They are going with Revisiting Bug Triage and Resolution Practices. In this paper they studied about bug triaging and fixing practices, including bug reassignments and reopenings, in the context of the Mozilla Core and Firefox projects, which they consider to be representative examples of a large-scale open source software project. Also they have plan to conduct qualitative and quantitative analysis of the bug assignment practices. We are interested in providing insights into several areas: triage practices, review and approval processes; root cause analysis of bug reassignments and reopens in open source software projects; and recommendations for improvements/redesign of bug tracking systems.

In [2] this paper, they introduce a graph model based on Markov chains, which captures bug tossing history. This model has several desirable qualities. First, it reveals developer networks which can be used to discover team structures and to find suitable experts for a new task. Second, it helps to better assign developers to bug reports. In our experiments with 445,000 bug reports, our model reduced tossing events, by up to 72%. In addition, the model increased the prediction accuracy by up to 23 percentage points compared to traditional bug triaging approaches.

In [3] recent research shows that optimizing recommendation accuracy problem and proposes a solution that is essentially an instance of content-based recommendation (CBR). However, CBR is well-known to cause over-specialization, recommending only the types of bugs that each developer has solved before. This problem is critical in practice, as some experienced developers could be overloaded, and this would slow the bug fixing process. In this paper, they take two directions to address this problem: First, we reformulate the problem as an optimization problem of both accuracy and cost. Second, we adopt a content-boosted collaborative filtering (CBCF), combining an existing CBR with a collaborative filtering recommender (CF), which enhances the recommendation quality of either approach alone.

In [4] Current techniques either use information retrieval and machine learning to find the most similar bugs already fixed and recommend expert developers, or they analyze change information stemming from source code to propose expert bug solvers. Neither technique combines textual similarity with change set analysis and thereby exploits the potential of the interlinking between bug reports and change

sets. In this paper, they present our approach to identify potential experts by identifying similar bug reports and analyzing the associated change sets. Studies have shown that effective bug triaging is done collaboratively in a meeting, as it requires the coordination of multiple individuals, the understanding of the project context and the understanding of the specific work practices. Therefore, they implemented approach on a multi-touch table to allow multiple stakeholders to interact simultaneously in the bug triaging and to foster their collaboration.

III. PROPOSED SYSTEM

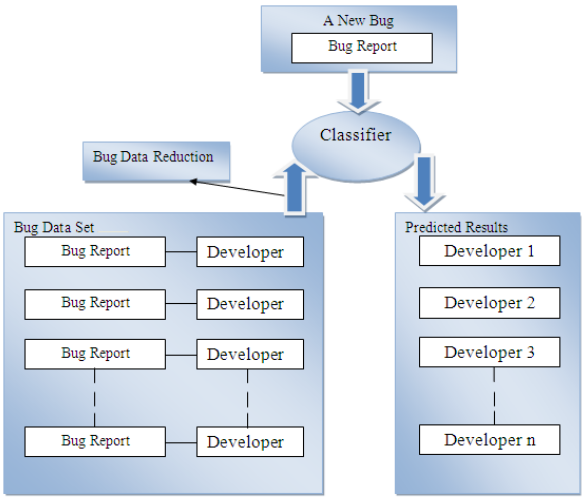

Fig.1 System Architecture

The diagram in figure 1 illustrated the system architecture of the proposed system. The input to the system is in the form of bug data set. The bug data set consists all the details of software bugs. Each bug has bug report and the details of the developer who have worked on that respective bug. The bug report is mainly divided in two parts, summary and description. The proposed system gives predicted results in form of output. Basically, there are two types of users in the proposed system. First is the developer and second is the tester. Developer will get software bugs assigned to him. Developer can work on only one software bug at a time. Tester can add new bugs to the system.

As shown in figure 1, the proposed system makes use of bug data reduction. In the proposed system, to save the labor cost of developers, the data reduction for bug triage is made. Bug daya reduction is applied in phase of data preparation of bug triage. Data reduction mainly has two goals. Firstly, reducing the data scale and secondly, improving the accuracy of bug triage.

Techniques of instance selection and feature selection are used for data reduction. Instance selection and feature selection are widely used techniques in data processing. For a given data set in a certain application, instance selection is to obtain a subset of relevant instances (i.e., bug reports in bug data) while feature selection aims to obtain a subset of relevant features (i.e., words in bug data). In the proposed system, the combination of instance selection and feature selection is used.

The proposed system will be implemented in java language so it will be platform independent. As there is no restriction on the size of bug's information, a tester can add large number of bugs in the system. This is one of the biggest advantages of the proposed system. Since all the bug's information is open to all the developers, it takes less time for the developer to take the decision. Developer can quickly choose the bug to fix.

Since bug triage aims to predict the developers who can fix the bugs, we follow the existing work to remove unfixed bug reports, e.g., the new bug reports or will-not-fix bug reports. Thus, we only choose bug reports, which are fixed and duplicate (based on the items status of bug reports). 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 our work, we remove the developers, who have fixed less than 10 bugs.

IV. CONCLUSION

Bug triage is an expensive step of software maintenance in both labor cost and time cost. The proposed system aims to form reduced and high-quality bug data in software development and maintenance. Data processing techniques like instance selection and feature selection are used for data reduction. The proposed system can be used for any open source projects that generate huge bug data. Various software companies working on projects like banking, food chain management can use the application of the proposed system.

REFERENCES

[1] Revisiting Bug Triage and Resolution Practices , Olga Baysal, Reid Holmes, and Michael W. Godfrey David R. Cheriton School of Computer Science University of Waterloo Waterloo, ON, Canada {obaysal, rtholmes, migod}@uwaterloo.ca.

[2] Improving Bug Triage with Bug Tossing Graphs Gaeul Jeong ∗ Seoul National University gejeong@ropas.snu.ac.kr.

[3] COSTRIAGE: A Cost-Aware Triage Algorithm for Bug Reporting Systems Jin-woo Park, Mu-Woong Lee, Jinhan Kim, Seung-won Hwang, POSTECH, Korea, Republic of, jwpark85,sigliel,wlsgks08,swhwang}@postech.edu

[4] Collaborative Bug Triaging using Textual Similarities and Change Set Analysis, Katja Kevic, Sebastian C. Muller, Thomas Fritz, and Harald C. Gall ¨ Department of Informatics University of Zurich, Switzerland katja.kevic@uzh.ch {smueller, fritz, gall}@ifi.uzh.ch.

[5] Automatic Bug Triage using Semi-Supervised Text Classification, Jifeng Xuan1 He Jiang2, 3 Zhilei Ren1 Jun Yan4 Zhongxuan Luo1, 2 1 School of Mathematical Sciences, Dalian University of Technology, Dalian, 116024 China 2 School of Software, Dalian University of Technology, Dalian, 116621 China 3 State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, 100190 China 4 Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing, 100190 China 1

.