ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

Vol. 4, Issue 11, November 2016

# A Survey on Cost-Aware Triage Technique for Improving Bug Triage

Chandana[1], T Ramamohan[2], J Raghunath [3]

Student, Dept. of CSE, GATES Institute of Technology, Affiliated to JNTUA University, Anantapur, India[1]

Assistant Professor, Dept. of CSE, GATES Institute of Technology, Affiliated to JNTUA University, Anantapur, India[2]

Assistant Professor, Dept. of CSE, GATES Institute of Technology, Affiliated to JNTUA University, Anantapur, India[3]

**ABSTRACT:** Lots of the application businesses desires to handle huge number of application bugs every day. Software bugs are unavoidable and fixing software bugs is an expensive task. The goal of powerful Trojan horse triaging application is to assign possibly experienced developers to new-coming computer virus studies. To cut down time and fee of trojan horse triaging, an automatic process is proposed in this project that predicts a developer with crucial experience to remedy or repair the brand new coming bug report. In this project, 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 program repositories, application of knowledge pre-processing, information management in malicious program repositories, computer virus information discount, characteristic determination, illustration, bug triage.

## I. INTRODUCTION

Many software companies spend most of the money in fixing the bugs. Giant software tasks have worm repository that collects all of the expertise related to bugs. In worm repository, each and every application worm has a malicious program file. The bug report consists of textual information regarding the bug and updates related to status of bug fixing.

As soon as a malicious program report is formed, a human triage assigns this bug to a developer, who will try to repair this malicious program. 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.

The method of assigning a proper developer for fixing the worm is called Trojan horse triage. Bug triage is one of the advanced steps in dealing with of bugs in software projects. Guide trojan horse triage through a human triage is time drinking and error-susceptible given that the number of every day bugs is significant and lack of advantage 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 tremendous scale data and secondly low excellent of knowledge. As a result of big number of every day mentioned bugs, the quantity of computer virus reviews is scaling up within the repository. Noisy and redundant bugs are degrading the first-class of bug experiences.

In this project 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] it is pointed out that worm triaging is an error-susceptible, tedious and time drinking project. They're going with Revisiting bug Triage and decision Practices. In this paper it is studied about worm triaging and fixing practices, including malicious program reassignments and reopening, in the context of the Mozilla Core and Firefox tasks, 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 considering supplying insights into a couple of areas: triage practices, review and approval procedures; root cause analysis of trojan horse reassignments and reopens in open source software projects; and recommendations for improvements/redesign of bug tracking systems.

In [2] they introduce a graph mannequin established on Markov chains, which captures trojan horse tossing history. This mannequin has a number of fascinating traits. First, it reveals developer networks which can be utilized to detect team structures and to seek out suitable specialists for a brand new venture. Secondly it helps to raised assign builders to trojan horse studies.

In our experiments with 445,000 bug studies, our mannequin decreased tossing activities, by using as much as 72%. In addition, the model increased the prediction accuracy by using up to 23 percentage points in comparison with average malicious program triaging methods.

In [3] contemporary study shows that optimizing recommendation accuracy drawback and proposes a solution to an illustration of content-founded suggestion. However, CBR is recognized to purpose over-specialization, recommending most effective the types of bugs that each and every developer has solved before. This quandary is principal in apply, as some skilled builders might be overloaded, and this would sluggish the worm fixing system.

In this paper, they took two instructions to address this situation: First, to reformulate the main issue as an optimization obstacle of both accuracy and price. Second to adopt a content material-boosted collaborative filtering (CBCF), combining a present CBR with a collaborative filtering recommender (CF), which boosts the recommendation excellent of either process alone.

In [4] Present systems either use understanding retrieval and desktop studying to search out essentially the most an identical bugs already fixed and advise expert developers, or they analyze exchange knowledge stemming from supply code to advocate educated computer virus solvers.

Neither manner combines textual similarity with trade set analysis and thereby exploits the abilities of the interlinking between worm reviews and change sets. An approach to identify potential experts by identifying similar bug reports and analyzing the associated change sets. Experiences have shown that powerful worm triaging is done collaboratively, as it requires the coordination of a couple of members, the figuring out of the project context and 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

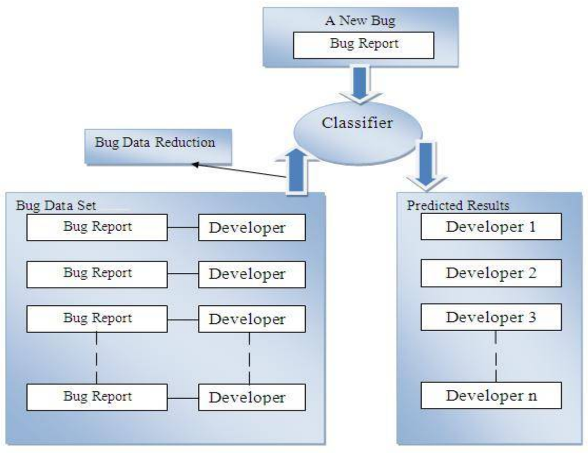

Fig.1 System Architecture

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 4, Issue 11, November 2016**

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 labour cost of developers, the data reduction for bug triage is made. Bug data reduction is applied in phase of datapreparation of bug triage. Data reduction mainly has twogoals. 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 labour 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 malicious program Triage and decision Practices , Olga Baysal, Reid Holmes, and Michael W. Godfrey David R. Cheriton school of computer Science university of Waterloo , ON, Canada obaysal, rtholmes, migod@uwaterloo.Ca.
2. Improving malicious program Triage with worm Tossing Graphs Gaeul JeongSeoul countrywide college gejeong@ropas.Snu.Ac.Kr
3. COSTRIAGE: A fee-mindful Triage Algorithm for bug Reporting methods Jin-woo Park, Mu-Woong Lee, Jinhan Kim, Seung-won Hwang, POSTECH, Korea, Republic of, jwpark85,sigliel,wlsgks08,swhwang@postech.Edu
4. Collaborative worm Triaging using Textual Similarities and change Set evaluation, 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 institution of technological know-how, Dalian, 116621 China 3 State Key Laboratory of pc Science, Institute of application, Institute of application, chinese Academy of Sciences,Beijing, 100190 China.
6. N. E. Fenton and S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, 2nd ed. Boston, MA, USA: PWS Publishing, 1998.
7. Y. Freund and R. E. Schapire, “Experiments with a new boosting algorithm,” in Proc. 13th Int. Conf. Mach. Learn., Jul. 1996, pp. 148– 156.
8. Y. Fu, X. Zhu, and B. Li, “A survey on instance selection for active learning,” Knowl. Inform. Syst., vol. 35, no. 2, pp. 249–283, 2013.
9. I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” J. Mach. Learn. Res., vol. 3, pp. 1157–1182, 2003.
10. M. Grochowski and N. Jankowski, “Comparison of instance selection algorithms ii, results and comments,” in Proc. 7th Int. Conf. Artif. Intell. Softw. Comput., Jun. 2004, pp. 580–585.
11. H. Brighton and C. Mellish, “Advances in instance selection for instance-based learning algorithms,” Data Mining Knowl. Discovery, vol. 6, no. 2, pp. 153–172, Apr. 2002.
12. 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.