International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No.03, April 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

# Clone detection and Bug Triage with Software Data Reduction Techniques

**Parmeshwar Pawar[1], Nitin Nandankar[2], Kiran Nikam[3], Swati Shirsat[4], Prof. Kopal Gangarde[5]**

[1,2,3,4,5]*Department Of Computer Engineering Genba Sopanrao Moze College of Engineering (Pune)*

**ABSTRACT**

*Bug triage is a fundamental step during bug fixing. Bug triage is the way toward fixing bug whose primary target is to accurately apportion a designer to another bug additionally taking care of. Many software organizations spend their majority of cost in managing these bugs. To reduce the time cost in manual work and to improve the working of programmed bug triage, two procedures are connected in particular content characterization and double arrangement. In writing different papers address the issue of information diminishment for bug triage, i.e., how to lessen the scale and enhance the nature of bug information. By joining the example determination and the component choice calculations to at the same time reduce the information scale and upgrade the correctness of the bug reports in the bug triage .According to writing, need to build up a powerful model for doing information lessening on bug information set which will decrease the size of the information and increment the nature of the information., by minimizing the time and cost. System produces bug report and assigns that bug to appropriate developer.*

***Keywords-Bug triage, data reduction, Instance selection, Feature selection, Data, Mining.***

## 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 [1]. Once a bug report is formed, a human trigger assigns this bug to a developer, who will try to fix this bug. This developer is recorded in an item.

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 [2]. 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 [3]. 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. The 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 [4].

Putting away points of interest of bugs, it assumes a critical part. Bug stores are broadly utilized for keeping up programming bugs, e.g. a well-known and open source bug store, Bugzilla. A recorded bug is known as a bug data, once a bug report is created, a human trigger doles out this bug to a designer, who will endeavor to settle this bug [2]. A designer, who is allocated to another bug report, begins to settle the bug in view of the information of chronicled bug settling. Bug triage is a routine of passing the settling bugs to correct engineer. An engineer, who is relegated to an imaginative bug data, begins to settle the bug based on the data of past bug settling. Regularly, the engineer pays endeavors to perceive the new bug report and generally settled bugs as a kind of perspective (e.g., hunting down comparative bugs and applying available answers for the new bug) [3]. Status of a bug report is changed by late consequence of taking care of this bug until the bug is totally settled. Displayed work utilizes the methodologies in view of content demeanor to help bug triage. In such a way, the rundown of a bug report are

concentrate as the printed content while the engineer who can join this bug is as the name for grouping. It gives low precision [4].

## II.LITERATURE SURVEY

**According to [1]** Bug triage is an expensive step of software maintenance in both labor cost and time cost. Here combine feature selection with instance selection to reduce the scale of bug data sets as well as improve the data quality. To determine the order of applying instance selection and feature selection for a new bug data set, to extract attributes of each bug data set and train a predictive model based on historical data sets. Empirically investigate the data reduction for bug triage in bug repositories of two large open source projects, namely Eclipse and Mozilla. The work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance.

Approach [2] Present a technique for finding faults in PHP Web applications that is based on combined concrete and symbolic execution. The work is novel in several respects. First, the technique not only detects runtime errors but also uses an HTML validator as an oracle to determine situations where malformed HTML is created. Second, address a number of PHP-specific issues, such as the simulation of interactive user input that occurs when user-interface elements on generated HTML pages are activated, resulting in the execution of additional PHP scripts. Third, to perform an automated analysis to minimize the size of failure-inducing inputs tags.

According to system [3] it reviews the problems with using simple K-Means in the classification of data sets. The effectiveness of Quad Tree based EM clustering algorithm in predicting faults while classifying a dataset, as compared to other existing algorithms such as, K-Means has been evaluated. The Quad Tree approach as signs appropriate initial cluster centers and eliminates the outliers. K-Means is considered to be one of the simplest methods to cluster data. However, the proposed EM algorithm is used to cluster data effectively. Combining the

Quad Tree approach and the EM algorithm gives a clustering method that not only its the data better in the clusters but also tries to make them compact and more meaningful. Using EM along with Quad Tree makes the classification process faster. With K-means, convergence is not guaranteed but EM guarantees elegant convergence.

The conducted case study on [4] high impact bugs, which classified bugs reported to four open source projects into six types of high impact bugs. In the case study, one hundred bug reports were manually inspected for each project and are classified into six types of high impact bugs based on previous studies which focus on high impact bugs. The case study aimed to reveal distributions of high impact bugs in reported bugs and overlapped relation ships among high impact bugs.

G. Jeong, S. Kim, and T. Zimmermann [5], author proposed bug tossing graph model can be easily incorporated into existing bug triaging systems. Find out that over 37 percent of bug reports have been reassigned in manual bug triage to other developers specifically in case of Mozilla and Eclipse. Proposed the model increased the prediction accuracy as compared to traditional bug triaging approaches.

Q. Shao, Y. Chen, S. Tao, X. Yan, and N. Anerousis [6], bug tossing is the same as ticket routing that is transferring a problem ticket among various expert groups in search of the right resolver to the ticket, which is a well-known problem in the machine learning literature. Most approaches use various statistical models to mine workflow from activity logs.

C. Sun, D. Lo, S. C. Khoo, and J. Jiang [7], in this paper author propose a retrieval function (REP) to identify such duplicates accurately between two bug reports. Proposed approach is twofold, first BM25F is an effective textual duplicates measure that is designed for short unstructured queries and seconds a new retrieval function REP fully utilizing text and other information available in reports such as product.

A. Srisawat, T. Phienthrakul, and B. Kijsirikul, [8], paper proposed SV-kNNC approach for data reduction to enhance performance of kNN. Proposed algorithm is three fold approaches, first support vector machines (SVMs) are applied to select some important training data then weights are allocate to each training instance based on k-mean clustering and finally classify the query instances by kNN classification process.

## III.PROBLEM STATEMENT

In this work to design and implement a system for software bug triaging and reporting. System first detect the different level code clone scenario and second identifying and tracing the bugs like different kind of exceptions into source code.

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No.03, April 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

**PROPOSED SYSTEM:**

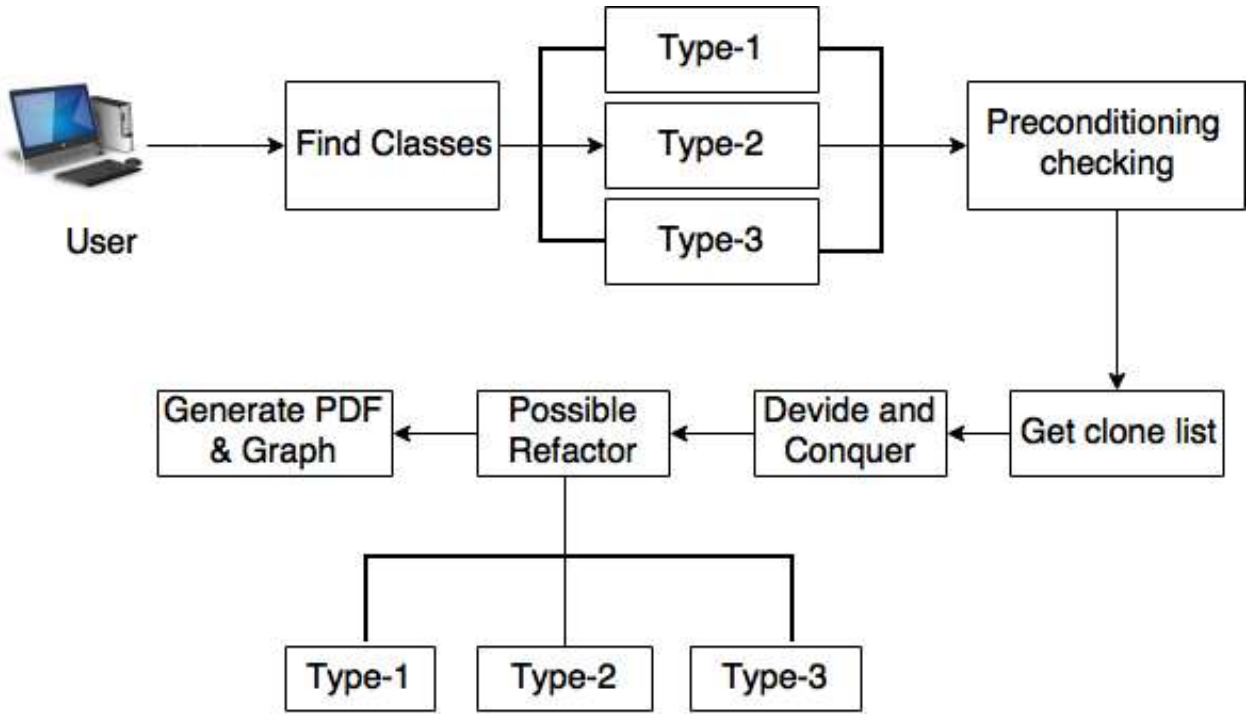



**Figure 1: Proposed system architecture**

**Data Reduction**

- Reducing the Data Scale - reduce scales of data sets to save the labor cost of developers.
- The reduced data set can be handled more easily by automatic techniques (e.g., bug triage approaches) than the original data set.

**Feature Selection**

- Replace the original data set with the reduced data set for bug triage.
- For a given data set in a certain application, instance selection is to obtain a subset of relevant instances.

**Clone Detection**

- First detect the multiple clones
- Find the re-factor possibility of all clones

**Bug Triage**

- Find the different type exception which is generate runtime environment.
- Create the bug report for system and notify to developer.

**ALGORITHM USED**

**1: Algorithm for find class level node detections**

**Input :** Java master class which contains many subclass M(s).

**Output**: Generate multiple tree buckets with class separation C[i]…[n].

**Step 1:** for each (read each node i to null )

**Step 2:** List Siblist= node i

**Step 3:** List NodePair= node i+1

**Step 4:** if (Siblist !=Nodepair)

New class node created with node i

Break;

**Else**

Continue with node i;

**Step 5:** Add C[node] <= node i

End for

**2: Algorithm for find edges from each tree node**

**Input :** each class as **TreeNode**

**Output** : Find all similar edges **Treenodelist** from **TreeNode**

**Arraylist=Treenodelist;**

**Step 1:** Read each Link from **Treenode**

**Step 2:** if **(Treenode.pattern (Linknodepattern))**

**Step 3: Treenodelist <= Treenode**

End for

**Step 4:** execute step 1 on all TreeNode

**3: Algorithm for finding clones from master class**

**Input :** Master class M with multiple subclass

**Output :** Classified clones viz **(Type1,Type2,Type2)**

**Temporary variables:** Reader, ArrayList T1, T2, T3

**Step 1:** read each line data= Reader. Line ()

**Step 2:** check pattern from (data)

if (clone.Node.mapped.single (data))

T1= data. Node

Else if (clone.Node.mapped.double (data))

T2=data. Node

Else if (clone.Node.mapped.tripple (data))

T3=data. Node

**Step 3:** consider as mapped node of (T1, T2, and T3)

**Step 4:** classify all clones as Type(1)…….Type(3)

**Step 5:** Show all clones

End for

**4: Algorithm for Precondition violation checking**

**Input** each class C with nodes

**Output :** Boolean 1 if violet else 0

**Step 1:** Read each line from data=reader.read(C)

**Step 2:** if(data.contains(break,control) || SubClass.Type || Param name diff || does not have void Type || continue, break || conditional(return) || Equals (break,continue))

Early stop, violation

**Step 3:** end for

**5: Algorithm A divide-and-conquer statement mapping process based on nesting structure.**

**Input:** two isomorphic NSTs

**Output:** the final mapping solution

**Step 1:** SetLength = NSTi.Length +NSTJ.Length

**Step 2:** **while** level SetLength !=0 **do**

**Step 3:** cpNodesi = nodes at level of NSTi

**Step 4:** cpNodesj = nodes at level of NSTj

**Step 5:** **for each** cpi 2 cpNodes i **do**

**Step 6:** SimScore =Mapping (NSTi, NSTj)

**Step 7:** if (SimScore ==0)

Count ++

End for

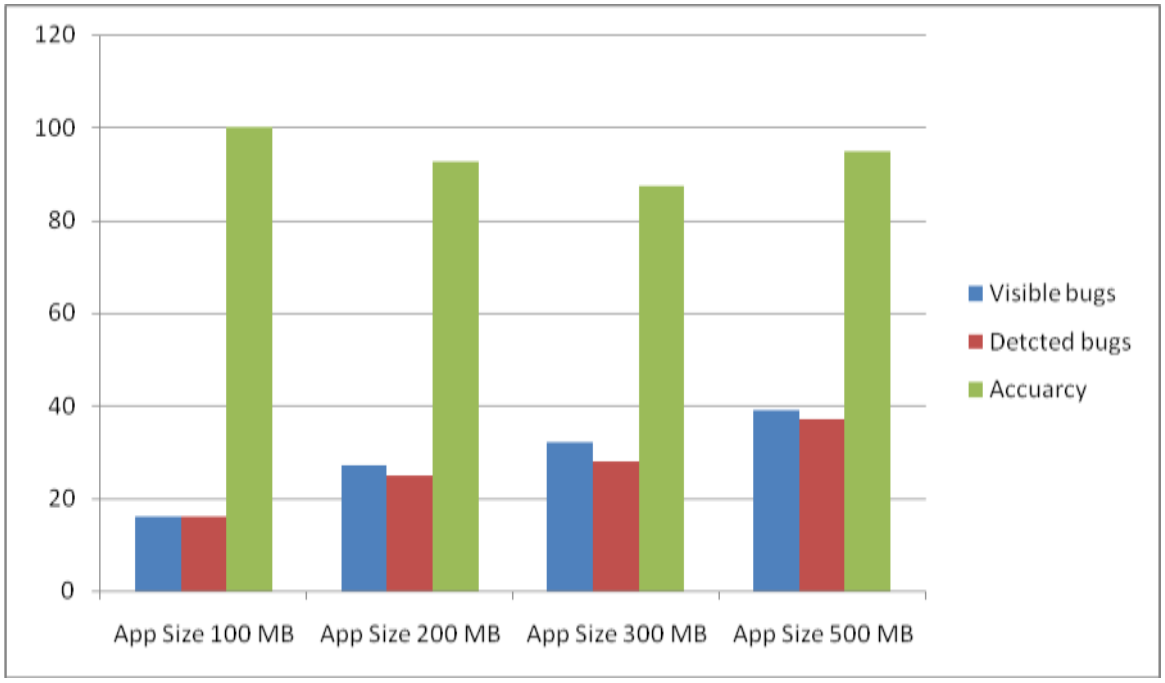
**Step 8:** if(Count>1)

Display to as refactoring possible.

**Step 9:** end procedure

## IV.RESULTS AND DISCUSSIONS

The below figure 2 shows the overall bug detection accuracy into the different application, when application size increased even accuracy cant be compromised.



## V.CONCLUSION

Bug triage is an expensive step of software maintenance in both labor cost and time cost. The combine feature selection with instance selection to reduce the scale of bug data sets as well as improve the data quality. To determine the order of applying instance selection and feature selection for a new bug data set, to extract attributes of each bug data set and train a predictive model based on historical data sets. The work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance. For predicting reduction orders, plan to pay efforts to find out the potential relationship between the attributes of bug data sets and the reduction orders. To propose bug data diminution to reduce the scale and to get better the quality of data in bug repositories. Which is applied as a stage in data training of bug triage. Here to mingle existing techniques of instance selection and feature selection to eliminate certain bug information and words. A crisis for reducing the bug data is to determine the order of applying instance selection and feature selection, which is denoted as the prediction of reduction orders. In this section, first how to apply instance selection and feature selection to bug data, i.e. data shrinking for bug triage. Then, to list the benefit of the data reduction. Those are Bug summery that will summarize the bug generated report in the form of pdf file. It will contain bug summery, bug deadline, bug description and suggestions by developer if any. Graphical representation for bug assigning and completion by developer is also provided. This makes the analysis easier for the higher authority to decide the developer to assign further bug to repair. In future work, to expect improving the eventual outcomes of data diminishment in bug triage to explore how to set up a first rate bug data set and handle a space specific programming task. For predicting reducing orders, plan to pay attempts to find the potential relationship between the characteristics of bug data sets and
the diminishment orders. In Future System used as:

- For IT industry to manage bug solution process task.
- Used automatic bug triage in industry.

## VI.FUTURE WORK

Need to implement this system for all high level supporting as well testing, Now current work is only applicable for Object oriented code. So need to improvement in future where it becomes for all languages.

## REFERENCES

[1] Jifeng Xuan, He Jiang, Yan Hu, Zhilei Ren, Weiqin Zou, Zhongxuan Luo, and Xindong Wu, IEEE VOL. 27, no. 1, January 2015

[2] . Kim, H. Zhang, R. Wu, and L. Gong, \Dealing with noise in defect prediction,"in Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng., May 2010, pp. 481490.

[3] G. Jeong, S. Kim, and T. Zimmermann, \Improving 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.

[4] Q. Shao, Y. Chen, S. Tao, X. Yan, and N. Anerousis, ticket routing by resolution sequence mining,"in Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2008, pp. 605 613.

[5] G. Jeong, S. Kim, and T. Zimmermann, “Improving bug triage with tossing graphs,” in Proc. Joint Meeting 12th Eur. Softw. Eng. Conf. 17th ACM SIGSOFT Symp. Found. Softw. Eng., Aug. 2009, pp. 111–120.

[6] Q. Shao, Y. Chen, S. Tao, X. Yan, and N. Anerousis, “Efficient ticket routing by resolution sequence mining,” in Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2008, pp. 605–613.

[7] C. Sun, D. Lo, S. C. Khoo, and J. Jiang, “Towards more accurate retrieval of duplicate bug reports,” in Proc. 26th IEEE/ACM Int. Conf. Automated Softw. Eng., 2011, pp. 253–262.

[8] A. Srisawat, T. Phienthrakul, and B. Kijsirikul, “SVkNNC: An algorithm for improving the efficiency of knearest neighbor,” in Proc. 9th Pacific Rim Int. Conf. Artif. Intell., Aug. 2006, pp. 975–979.