International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 02 | Feb-2016 www.irjet.net p-ISSN: 2395-0072

# A SURVEY ON BUG TRACKING SYSTEM FOR EFFECTIVE BUG CLEARANCE

**M.Suresh[1], M.Amarnath[2], G.Baranikumar[3], M.Jagadheeswaran[4]**
[1]Assitant professor, Departement of Information Technology, mail4sureshuni@gmail.com.
[2, 3, 4] B.TECH, 4th year student, Departement of Information Technology, Manakula Vinayagar Institute of Technology,Pondicherry,India.
[2]amar.moorthy1994@gmail.com,[3]baranikum2r@gmail.com,[4]jagan.mohan239864@gmail.com.

---

***Abstract:*** *Bugs are important challenge for a software organization. Software organization spends over 45 percentages of their resources in handling these bugs. Managing these bugs manually are difficult and error prone. So an automatic approach of instance selection and feature selection method is combined to handle the bugs, then the bugs are distributed to bug solving experts. An inevitable step in fixing the bugs is assigning a bug solving expert. The problem is majority of bugs are assigned to experts who has less experience in that domain which can leave the bugs unsolved. So using term selection method a bug solving expert is predicted automatically depending upon the type of bugs. A history of these cleared bugs is maintained using historical data management system. This automatically resolves a bug which is reported and solved in prior. This highly reduces the time and cost involved in the bug clearance.*

**Keywords:** Bug data reduction, Instance selection, feature selection, Historical data management system.

## 1. INTRODUCTION

Bug is important challenge for any software organization. Most of the software companies need to deal with large number of software bugs every day. Software bugs are unavoidable and fixing software bugs is an expensive task [2]. In fact Software organization spends most of their resources in handling these bugs. For managing software bugs, bug repository plays an important role. In software development and Maintenance, a bug repository is a significant software repository for storing the bugs submitted by users. Most of the software which is open source projects has an open bug repository which allows developers as well as users to submit issues or defects in the software that suggest possible solutions and remark on existing bug reports. The drawback is that large-scale software projects are so much large that makes the triaging process very difficult. The inefficient data and unclear data add redundancy data to the data repository and create a great challenge to the software experts [1]. 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 [2].

A bug repository gives a data stage support about types of tasks on bugs, e.g., fault prediction, bug localization, and reopened bug analysis. Large software projects convey bug repositories (also called bug tracking systems) support information collection and to assist developers to handle bugs [3].

The number of regular occurring bugs for open source large-scale software projects is so much large that makes the triaging process very difficult and challenging[1]. 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 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 triage 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 [2].

Before verifying and modifying a bug, each bug report must be assigned to a relevant developer who could fix it. In traditional bug repositories, all the bugs are manually triaged by some specialized developers. Aiming to reduce the human labor costs, some supervised text classification approaches have been proposed for automatic bug triage. After that the nature of the bugs is predicted using a predictive algorithm and then predict relevant developers for the incoming bug reports with these classifiers.

A bug repository gives a data stage support about types of tasks on bugs, e.g., fault prediction, bug localization, and reopened bug analysis. Large software projects convey bug repositories (also called bug tracking systems) support information collection and to assist developers to handle bugs [3].

The number of regular occurring bugs for open source large-scale software projects is so much large that makes the triaging process very difficult and challenging[1]. 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 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 triage 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 [2].
Before verifying and modifying a bug, each bug report must be assigned to a relevant developer who could fix it. In traditional bug repositories, all the bugs are manually triaged by some specialized developers. Aiming to reduce the human labor costs, some supervised text classification approaches have been proposed for automatic bug triage. After that the nature of the bugs is predicted using a predictive algorithm and then predict relevant developers for the incoming bug reports with these classifiers.

## 2. TECHNIQUES USED

### 2.1 Top-K pruning algorithm

Top-K ranking query in uncertain databases aims to find a top K tuples according to a ranking function. The interplay between score and uncertainty makes top-K ranking in uncertain databases an intriguing issue, leading to rich query semantics. Recently, a unified ranking framework based on parameterized ranking functions (PRFs) is formulated, which generalizes many previously proposed ranking semantics. Under the PRFs based ranking framework, efficient pruning approach for Top-K ranking on dataset with tuple uncertainty has been studied. However, this cannot be applied to top-K ranking on dataset with value uncertainty (described through attribute-level uncertain data model), which are often natural and useful in analyzing uncertain data in many applications.

### 2.2 Instance selection and feature selection

Instance selection and feature selection algorithm deals with clustering the similar bug data sets. . Feature and Instance Selection belong to the practice of data preparation (or pre-processing), which is a preliminary process that transforms raw data into a format that is convenient to the data mining (or machine learning) algorithm. Usually, data is stored in a table-like format: the columns of these tables are the attributes or features - they describe the data - and the rows, or lines, are the records or instances - they are the examples of the concept stored in the data. Feature and Instance selection processes allow applications, such as classification or clusterization, to focus only on the important (or relevant) attributes and records to the specific concept that is in study. By removing noise, irrelevant and redundant features and instances, and reducing the overall dimensionality of a dataset, feature and instance selection have been demonstrated to improve the performance of most machine learning algorithms, speed up the output of models and allow algorithms to deal with datasets whose sizes are gigantic.

Feature selection algorithms perform very differently in identifying and removing irrelevant, redundant and randomly class-correlated features. Most of the works on instance selection have been based on Nearest Neighbor classification which finds a subset such that every member of the original dataset is closer to a member of the subset of the same class than to a member of the subset of a different class.

### 2.3 Apollo's algorithms

Apollo generates test inputs for a web application, monitors the application for crashes, and validates that the output conforms to user. Apollo algorithm, a new sensor information processing tool for uncovering likely facts in noisy participatory sensing data. Apollo belongs to a category of tools called fact-finders. It is the first fact-finder designed and implemented specifically for participatory sensing. When data tuples are clustered and ranked by Apollo, the quality of reported observations increases considerably. Apollo algorithm illustrates a fact-finding tool designed to uncover most likely truth in noisy participatory sensing data. Filtering the amount of data for correctness is an important challenge, commonly known in machine learning and knowledge discovery literature as fact-finding. Apollo is the first fact-finder designed specifically for participatory sensing data. The algorithm has the following reconfigurable parts: The parser: It uses a configuration file (that describes the format of the input data stream) to convert input data into a standard JSON format. Conceptually, the data stream is composed of source, observation tuples, where source identifies the data source (e.g., the Twitter user ID), and the observation content may be structured, as in the case of phones sharing sensor values, or unstructured, as in the case of people sharing text. The configuration file describes the observation format. The distance metrics: A library of different distance metrics is provided for clustering of observations. For unstructured text, these metrics reflect text similarity. For structured data, metrics compute differences in data vectors. Data can be multidimensional. For example, when cell-phones report sensor values at given locations, both measurements and locations can be elements of the data vector.

### 2.4 Naïve bayes classifier

The approach iteratively labels numerous unlabeled bug reports and trains a new classifier with labels of all the bug reports. A weighted recommendation list is

maintained to boost the performance by imposing the weights of multiple developers in training the classifier. Naive bayes classifiers are a family of simple probabilistic classifiers based on applying bayes theorem with strong (naive) independence assumptions between the features. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness and diameter features.

### 2.5 Markov chains method

A Markov chain is a random process that undergoes transitions from one state to another on a state space. It must possess a property that is usually characterized as "memorylessness": the probability distribution of the next state depends only on the current state and not on the sequence of events that preceded it. This specific kind of "memorylessness" is called the Markov property. Markov chains have many applications as statistical models of real-world processes. It reveals developer networks which can be used to discover team structures and to find suitable experts for a new task.

### 2.6 Content-based recommendation (cbr) and content based filtering (cbf)

It recommends only the types of bugs that each developer has solved before. In a content-based recommender system, keywords or attributes are used to describe items. To provide content-based recommendation we treat the prediction task as a text-categorization problem. Content based filtering combines cbr with a collaborative filtering recommender (cf), it identify potential experts by identifying similar bug reports and analyzing the associated change sets.

### 2.7 Defect prediction method

Defect prediction is a software task in software metrics, which is to predict whether a software artifact (e.g., A source code file, a class, or a module) contains a defect or not First, a data set of software modules is collected; second, each module is labeled (i.e., Whether a module contains defects) and the attributes of this module are extracted as metrics; third, a classifier (e.g., A decision tree) is trained as a predictive model; fourth, the trained classifier is used to predict whether a new module contains defects.

## 3. LITERATURE REVIEW

### 3.1 Towards Effective Troubleshooting with Data Truncation

Towards Effective Troubleshooting with Data Truncation deals with reducing the data present in the bug repository and improve the quality of data then reduce time and cost of bug triaging, it represent an automatic approach to predict a developer with relevant experience to solve the new coming report.. The bug data sets are obtained and techniques such as instance selection feature selection are applied simultaneously. The top k pruning is applied for improving results of data reduction quality, obtaining domain wise bug solution. Instance selection is for obtaining a subset of relevant instances (i.e., bug reports in bug data) .It is used to Remove noise and redundant instances,Remove non-representative instances.
Feature selection which aims to obtain a subset of relevant features (i.e., words in bug data) ,Sorting of words according to feature values .Top-K pruning algorithm for improving results of data reduction quality.

### 3.2 Technique to Combine Feature Selection with Instance Selection for Effective Bug triage

A Technique to Combine Feature Selection with Instance Selection for Effective Bug Triage .It addresses the issue of data reduction for bug triage by text classification techniques. Conventional software analysis is not totally suitable for the large-scale and complex data in software repositories. Data mining has developed as a promising means to handle software data. There are two difficulties related to bug data that may influence the effective use of bug repositories in software development tasks, namely the huge scale and the low quality. Therefore unfixed bugs are deleted from the bug repositories.

### 3.3 Automatic Bug Triage using Semi-Supervised Text Classification

Automatic Bug Triage using Semi-Supervised Text Classification propose a semi-supervised text classification approach for bug triage to avoid the deficiency of labeled bug reports in existing supervised approaches. This approach combines naive bayes classifier and expectation maximization to take advantage of both labeled and unlabeled bug reports. This approach trains a classifier with a fraction of

labeled bug reports. Then the approach iteratively labels numerous unlabeled bug reports and trains a new classifier with labels of all the bug reports. Then it employs a weighted recommendation list to boost the performance by imposing the weights of multiple developers in training the classifier. Before training a supervised classifier for bug triage, a necessary step is to collect numerous labeled bug reports, which are bug reports marked with their relevant developers.The semi-supervised text classification approach to improve the classification accuracy of bug triage. This semi-supervised approach enhances a NB classifier by applying expectation-maximization (EM) based on the combination of unlabeled and labeled bug reports. First, this semi-supervised approach trains a classifier with labeled bug reports. Then, the approach iteratively labels the unlabeled bug reports and trains a new classifier with labels of all the bug reports. To adjust bug triage, we update a semi-supervised approach with a weighted recommendation list (WRL) to augment the effectiveness of unlabeled bug reports. This WRL is employed to probabilistically label an unlabeled bug report with multiple relevant developers instead of a single relevant developer.

## 3.4 Reducing Features to Improve Bug Prediction

Recently, machine learning classifiers have emerged as a way to predict the existence of a bug in a change made to a source code file. The classifier is first trained on software history data, and then used to predict bugs. Two drawbacks of existing classifier-based bug prediction are potentially insufficient accuracy for practical use, and use of a large number of features. These large numbers of features adversely impact scalability and accuracy of the approach. Reducing Features to Improve Bug Prediction aims in classifier to first trained on software history data, and then used to predict bugs. The disadvantage of the traditional method is that, classifier-based bug predictions are potentially insufficient accuracy for practical use, and use of a large number of features. The system uses Naive Bayes and Support Vector Machine (SVM).The system mainly Gain Ratio for feature selection, along with the characterization of bug prediction results achieved when using feature selection. This paper proposes a feature selection technique applicable to classification-based bug prediction. This technique is applied to predict bugs in software changes, and performance of Naive Bayes and Support Vector Machine classifiers is characterized. . These features include everything separated by whitespace, in the code added or deleted in a change. This leads to a large number of features, in the thousands, and low tens of thousands. For larger project histories which span thousand revisions or more, this can stretch into hundreds of thousands of features. The addition of many non-useful features reduces a classifier's accuracy. Additionally, the time required to perform classification increases with the number of features, rising to several seconds per classification for tens of thousands of features, and minutes for large project histories. A standardapproach (in the machine learning literature) for handling large feature sets is to perform a feature selection process to identify that subset of features providing the best classification results. This paper introduces a feature selection process that discards features with lowest gain ratio until optimal classification performance is reached for a given performance measure.

## 3.5 Efficient Bug Triaging Using Text Mining

Efficient Bug Triaging Using Text Mining aims for an automatic approach to predict a developer with relevant experience to solve the new coming report. The techniques used are five term selection method. Term selection methods are used to reduce the high dimensionality of term space by selecting the most discriminating terms for the classification task. The methods give a weight for each term in which terms with higher weights are assumed to contribute more for the classification task than terms with lower weights. The goal of bug triaging is to assign potentially experienced developers to new-coming bug reports. To reduce time and cost of bug triaging, we present an automatic approach to predict a developer with relevant experience to solve the new coming report. It investigate the use of five term selection methods on the accuracy of bug assignment. In addition, it re-balance the load between developers based on their experience. It conduct experiments on four real datasets. To reduce the time spent triaging, it present an approach for automatic triaging by recommending one experienced developer for each new bug report. This information can help to manage the progress of these projects. In the last decade, practitioners have analyzed and mined these software repositories to support software development and evolution. One of the important software repositories is the bug tracking system (BTS). Many open source software projects have an open bug repository that allows both developers and users to submit defects or issues in the software, suggest possible enhancements, and comment on existing bug reports. It formulate the bug triaging process as a classification task where instances represent bug reports, features represent the terms of the report, and the class label represents the developer who fixed this report. This approach can help the triage process in two ways: 1) it may allow a triager to process a bug more quickly, and 2) it may allow a triager with less knowledge about systems and developers to perform bug assignments more accurately.

| Sl. no | Title | Year | Tools | Application Area | Pros | Cons | Analysis |
|---|---|---|---|---|---|---|---|
| 1 | Towards Effective Troubleshooting With Data Truncation | 2015 | Instance selection and feature selection | Bug Repository Handling. | The problem of handling huge number of data in bug repository is minimized. | Instance selection and feature selection is not completely enough to handle the data in bug repository. | Additionally, text processing and data processing algorithm must be used to handle the data in bug repository. |
| 2 | A Technique to Combine Feature Selection with Instance Selection for Effective Bug Triage | 2015 | Instance selection and feature selection | For an efficient bug triage. | It analysis data by considering the word dimension and bug dimension which helps in reducing duplicate and unnecessary bugs. | The order of applying instance selection and feature selection is not clearly explained which leads to inefficient system. | The order of applying instance selection and feature is selection must be determined by a predictive algorithm. |
| 3 | Automatic Bug Triage using Semi-Supervised Text Classification | 2010 | Naïve bayes classifier | Software bug handling. | It labels the bug data iteratively. The weighted list maintained, helps to boost the results obtained. | It only focuses on classifying the bugs in bug repository. The major problem in bug handling is that huge number of data in bug repository. | Techniques such as IS and FS must be used to reduce the data in bug repository. Proper predictive algorithm must be used to classify bugs. |
| 4 | Reducing Features to Improve Bug Prediction | 2009 | Support vector machine | Software development committee. | Provides a layout for minimizing techniques used in bug data reduction. | Minimizing the techniques used leads in less accuracy. | Necessary algorithm such as IS and FS must be used. |
| 5 | A Survey Paper on Efficient | 2015 | Instance selection feature | Automatic bug triaging. | It uses text classification techniques to | Too many repositories are used to handle | A comparator can be used |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Approach of Data Reduction Techniques for Bug Triaging System | | selection and historical data sets | | reduce the data in bug repository. Then uses predictive model to predict a developer based on historical data sets. | the bugs in bug repository which can be minimized. | to compare the new bug report with historical data sets. |
| 6 | A survey on bug-report analysis | 2015 | Bugzilla | Explains various techniques in bug handling | Research and explain background details about bug handling which makes bug handling an efficient task. | The accuracy obtains by this techniques are not more enough in handling the bugs. | Several more techniques such as instance selection and feature selection must be used to handle the bugs. |
| 7 | Automatic bug triage using text cateogarization | | Naïve bayes classifier | Large open source projects. | Have explained naïve bayes classifier for bug triaging. And investigated bug data sets efficiently for bug handling. | There is a chance of assigning the bugs to un-suited developers. Therefore the techniques used are inefficient. | Developers should be involved in bug triaging for an efficient bug handling. |
| 8 | Bug report network: varieties strategies and impact in Fos development | 2004 | - | Software development community. | It focuses on developing a model that provides enhanced software management techniques to handle bugs. | It just handles the bugs from single community. | Tools must be introduced to handle BRN's complexity. |
| 9 | Coping with an open bug repository. | 2004 | Bugzilla | Development of eclipse and Firefox projects. | Explained briefly about various challenges that are faced in handling bugs in bug repository. | It does not explained clearly about how overcoming these challenges. | After analyzing the scenario proper tools must be suggested for bug handling. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 10 | Defect tracking system | 2007 | CDTS and BTS | Development of projects in MNC's. | CDTS helps in keep tracking of defect, flaws, errors that can happen in development of a product. BTS provides a list of developers to solve the reported bugs. | CDTS does not provide a view of methodologies to handle the reported bugs. The process is time consuming when CDTS is combined with BTS. | Alternate approach must be used in handling these defects in the products. |
| 11. | Modeling bug report quality | 2007 | Bugzilla | Used as filter in Mozilla and Firefox project to resolve the bug. | It will reduce overall maintenance cost. Leads to better precession and recall results. | The drawback is that it will consider only a two projects. | The reported should provide complete guide about the bug. |
| 12. | Towards the next generation of bug tracking system | 2008 | Card sort technique | Software development project. | It will come out with useful recommendation that improve the working of BTS. | Though it provides more recommendation it does not provide any tool. | The feature that can implemented to good quality of bug report in BTS. The bug report that provides complete information that reporter gets an idea on how to file a bug report. |
| 13. | Duplicate bug report considered harmful really?? | 2008 | Infozilla | Software maintenance of large projects. | That all duplicate are not always considered to be harmful sometimes it provides additional information's. | The study was based on the survey results of bug report from eclipse project5. | BTS should consist a tool that scans the entire duplicate bug and extracts all the additional information that might be help in |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | fixing the bug. |
| 14. | What makes a good bug report | 2008 | cuezilla | Software development projects and product manufacturing. | It will mention the crucial role in resolving the bug. Identifies the why developer takes more time. | This will consider the response of experienced developers. It involves the self selection principles. | To improve the quality of bug report is that it should be made mandatory for all reporters to provide all relevant information required |
| 15. | New feature for duplicate bug detection | 2014 | Weka | Large open source projects and android development projects. | Undefined the use of automatically grouping the duplicate bug report. | Study shows that 36% of bugs are duplicate but in experiments only 1452 were duplicate. | The attributes should be further explored so that better duplicate detection system could be developed. |

**Table -1: Analysis of various bug tracking system**

## 4. Conclusion

The amount of data available in the bug repository plays an important role in bug handling.So as to reduce the data in bug repository, the bug reduction techniques must be implemented. The data in bug repository is majorly reduced by neglecting the redundancy data in the bug repository. Then the subset of the data in data repository is obtained. In future we planned to develop a predictive model which predicts a developer based on the type of bugs obtained. Then an historical data management system is maintained which keeps record of bugs which are reported and resolved in prior. A comparator checks with this historical data management system when a new bug is reported, if it is found to be reported again the historical data management system resolves it automatically without assigning to an expert.

## References

[1] B Jifeng Xuan, He Jiang, Yan Hu, Zhilei Ren, Weiqin Zou, Zhongxuan Luo, 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] karishmaMusale, Gorakshanath Gagare, "Towards Effective Troubleshooting With Data Truncation" , "International Journal of Advanced Research in Computer and Communication EngineeringVol. 4, Issue 11, November 2015"

[3] Ashwini Jadhava, Komal Jadhava, "A Survey on Software Data Reduction Techniques for Effective Bug Triaging", "International Journal of Computer Science and Information Technologies, Vol. 6 (5) , 2015, 4611-4612".

[4] JifengXuan, He Jiang, "Automatic Bug Triage using Semi-Supervised Text Classification", "Chinese Academy of Sciences, Beijing, 100190 China".

[5] Suvarnaa Kale, Ajay Kumar Gupta, "A Technique to Combine Feature Selection with Instance Selection for Effective Bug Triage", "International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064".

[6] 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.,2012

[7] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?," Proc. Intl. Conf. Softwarre Engineering (ICSE 06), ACM, 2006, pp. 361-370.

[8] N. Betten burg, R. Prem raj, T. Zimmermann, and S. Kim, "Reducing Features to Improve Bug Prediction," Proc. IEEE Conf. Software Maintenance (ICSM 08), IEEE Computer Society, Sep. 2008, pp. 337-345.

[9] J. Anvik , "Automatic bug reporting assignment, " in Proc 28th International Conference on Software Engineering. ACM, 2006, pp. 937–940.

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

[11] K. Baloga, L. Azzopardi, and M. de Rijke, "Formal models for expert finding in enterprise corpora," in Proc. 29th Annu. Int. ACMSIGIR Conf. Res. Develop. Inform. Retrieval, Aug. 2006, pp. 43–50.

[12] C. Aggarwaal and KP. Zhaao, "Towards graphical models for textprocessing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.

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

[14] S. Artzi, A. Kie_zunn, J. Dollby, F. Tip, D. Dig, A. Paradkar, and M. D.Ernst, "Findingbugs in web applications using dssynamic test generationand explicit-state model checkin," IEEE Softw., vol. 36,no. 4, pp. 474–494, Jul./Aug. 2010.

[15] D. Matter , A. Kuhan , and O. Nierstrasz , "Assigning bug reports using a vocabulary-based expertise model of developers," in 6th IEEE International Working Conference onMining Software Repositories, 2009, pp. 131–140

[16] D. Cubranic and G. C. Murphy , "Automatic bugs triaging usings text categorization," in Proc Sixteenth International Conference on Software Engineering, Citeseer, 2004, pp. 92–97.

[17] Jifeng Xuan , Hee Jiang , " Towards Effective Bug Triage", "IEEE transactions on journal 2013",pp.251-269 .