International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 11 (2018) pp. 9457-9459

Code Quality Measurement and Automatic Bug Triage Using Data Reduction Techniques

D.kiranmayi, K.Nasaramma, M.Bangaru.Lakshmi, G.Prasanna Priya

Assistant Professor
Vignan's Institute Of Information Technology,
Visakhapatnam, India.

Abstract

In the software development process software companies spend quality of the time for dealing with software bugs. An unavoidable step of fixing bugs is bug triage, which main target to allocate developers to a new bug. The process of bug triage is helps to trim down the schedule time and cost of the project. In addition to bug triage process, the project also applied classification technique to conduct automatic bug triage and generating bug report.

This paper addresses to generate quality of code and dealing with software bugs and bug triage using data reduction techniques. Data preprocessing methods namely instance selection, feature selection, extraction attributes applied on historical bug data sets. It helps to generate better predictive model.

In this work implemented predictive model as Lib SVM, which can classify the bug dataset. The new bug is added to the related bug class as model classifier. The entire work helps to the software development in terms of cost and time parameters.

Keywords: code quality metrics, code readability, data reduction techniques, data preprocessing

EXISTING SYSTEM

In existing system, we have noticed the problem of data reduction for bug triage, i.e., to reduce the bug dataset in order to save the human triage, cost of developers and improve the quality of the process in bug triage. Data reduction for bug triage aims to build a high-quality set of bug data by removing bug reports and unwanted words, which are redundant or non-informative.

Traditional software analysis is not completely suitable for the large-scale and complex data for development of quality in software systems. In traditional software development, new bugs are manually triaged by an expert developer, i.e., a human triage. Due to large number of bugs daily and lack of expertise of all the bugs, manual bug triage is expensive in terms of time, cost and in accuracy.

A time-consuming step of handling software bugs manually, In manual bug triage the percent of bugs are assigned by mistake, the drawback of the manual bug triage is to overcome in the proposed system with the help of the data mining techniques. Those techniques are data reduction, feature selection and instance selection, classification. In existing work has proposed an automatic bug triage approach, which applies text classification techniques to predict developers for bug reports and by using data reduction techniques generated a quality bug data sets. In this approach, a bug report is mapped to a document and a related developer is assigned to fix a particular bug. Then, bug triage is converted into a problem of generating bug report and is automatically solved with text classification techniques, e.g., Naive Bayes. (Based on the results of text classification, a human triage assigns new bugs).

To improve the accuracy of text classification techniques for bug triage, some new techniques are investigated, e.g., a tossing graph approach and a collaborative filtering approach. However, large-scale and low-quality bug data in bug repositories block the techniques of automatic bug triage. .Since software bug data are a kind of free-form text data (generated by developers), it is necessary to generate well-processed bug data to facilitate the application.

In Existing system only manual bug triage is addressed. When a new bug arrives the developer needs to assign the bug status manually. Some bug contains large amount of data while entering manual state it consumes lot of time as well as cost. For overcoming the problem of manual entering of a new bug we can propose automatic bug triage and additionally we can propose code quality metrics for generating bug and add data preprocessing techniques for bug dataset.

PROPOSED SYSTEM

In our work, we had combined existing techniques of instance selection and feature selection to simultaneously reduce the bug dimension and the word dimension. The reduced bug data contain bug reports than the original bug data and provide similar information over the original bug data. We evaluate the reduced bug data according to the scale of a data set and the accuracy of bug triage.

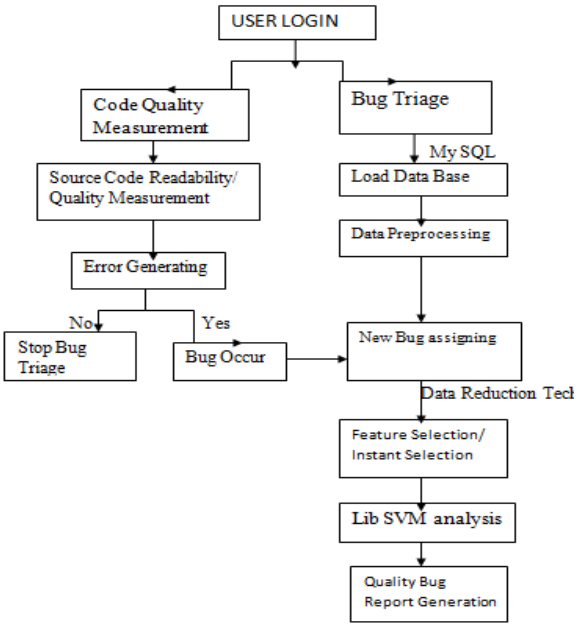
The proposed work addresses to generate quality of code and dealing with software bugs and bug triage using data reduction techniques. In the proposed system applied feature selection and instance selection for identifying the correct features from the dataset. As the feature selection influences the accuracy of any performance model. It is to study effectiveness of the performance model in connection with feature selection techniques. Here in this project Attribute Subset selection technique is used for extracting the attributes from the historical bug data. As a result dimensionality of the

module expands the number of features increase. Finding this attribute Subset Selection is also implemented.

By extracting the attributes from historical bug data sets and build a predictive model for a new bug data set, we propose a combination approach to resolve the problem of data reduction by applying data preprocessing techniques. This can be viewed as an application of instance selection and feature selection in bug repositories as a history of the bug dataset. We build a lib SVM classifier to display the order of applying instance selection and feature selection. As per the proposed model instance selection and feature selection improves the volume of historical bug dataset and data quality. 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. And when a new bug arrives we can directly enter the new bug in the historical bug data where in existing system only manual entry is available

SVM analysis for required bug dataset can classify the Bug status. In this project we build the process bug life cycle used. Some statuses are associated with bugs. When a new bug arrives that occurred and yet to be valid as new bug same as for (assigned bugs, active bugs, tested bugs, closed bugs, reopened bugs, Duplicate bugs). The various statuses used for the bug during bug life cycle. In various states defect goes through in its life cycle. This is how bug data manages in bug repositories

SYSTEM ARCHITECTURE



CODE QUALITY MEASUREMENT:

1. When the Code quality measurement button the action performed directs to SLOC Frame.
2. SLOC Frame constructors we will initialize the form elements.
3. Browse and select the file in the text field that you want to measure quality.
4. Click on the submit button the action performed method calls the methods in LOC class
5. From LOC get the values like total lines, packages, classes.
6. Call the new NewFrame() constructer initialize the form elements.
7. In NewFrame Class we will browse and select the file and generate the bugs and description.
8. Assign the developer to the new bug in bug03 class by giving the required details.
9. Insert bug1 and bug 2 values into the database.
10. View the employee details by retrieving the bug2 table.
11. By click next button we wil; redirect to bug2 class
12. Select the type of data from dropdownlist
13. Get Featured Selection dataset in bug2 class.
14. Select the type of bug report and get the bug report.
15. Submit the bug status details.
16. Get the rectified and new assigned bug history.
17. Generate the report in the graph format.

BUGTRIAGE ALGORITHM:

1. Click the Bugtriage and Assignment button and action method calls new dataset() constructor.
2. In Dataset Class
   a. Click browse to selct the file and read the data in the file.
   b. Click the view to retrieve the dataset from bug1 table.
   c. Click the Load to truncate the data in bug1 and load the dataset with values.
   d. Click next button to call preprocess class.
3. Preprocess the data in preprocess class
4. Assign the developer to the new bug in bug03 class by giving the required details.
5. Insert bug1 and bug 2 values into the database.
6. View the employee details by retrieving the bug2 table.
7. By click next button we will; redirect to bug2 class
8. Select the type of data from dropdown list
9. Get Featured Selection dataset in bug2 class.
10. Select the type of bug report and get the bug report.

11. Submit the bug status details.
12. Get the rectified and new assigned bug history.
13. Generate the report in the graph format.

CLASSIFIER ALGORITHM:

1. Use classifier to know which developer we need to assign to a bug.
2. Retrieve the details from history data.
3. Use the text classification technique to classify the words in the bug triage
4. Use feature selection or Instance selection to reduce data scale in bug dataset.
5. Feature selection: obtain a subset of relevant features by removing the uninformative words in the bug report.
6. Instance selection: is used to obtain a subset of relevant instances.
7. Combining both these algorithms we get a reduced bug data set and replace it with the original bug data.
8. Sent this to classifier.
1. 9.When a new bug is generated classifier assigns it a new developer.

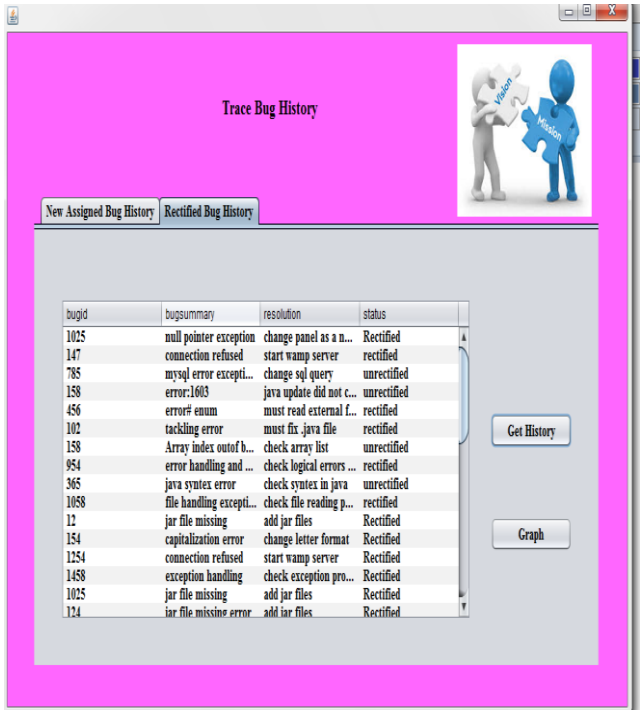


Figure 1: History of the Newly Assigned Bugs and Rectified and Unrectified Bugs

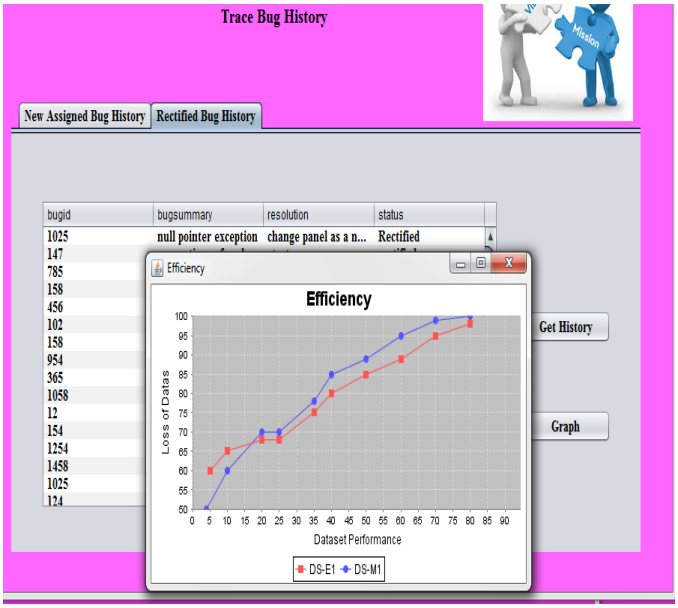


Figure 2: Performance and efficiency calculated

REFERENCES

[1] Jifeng Xuan, He Jiang, 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] Claire Le Goues, Westley Weimer Measuring Code Quality to Improve Specification Mining IEEE Transactions on Software Engineering Volume: 38, Issue: 1, Jan.-Feb. 2012.

[3] J. Jayashri Gholap1 , N. P. Karlekar2 Survey on Bug Triage with Software Data Reduction Techniques"IJSR Volume 4 Issue .12, December 2015

[4] C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing,"Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.

[5] Bugzilla, (2014). [Online]. Avaialble: http://bugzilla.org/

[6] K. Balog, L. Azzopardi, and M. de Rijke, "Formal models for expert finding intenterprise corpora," in Proc. 29th Annu. Int. ACM SIGIR Conf. Res.Develop. Inform.Retrieval, Aug. 2006, pp. 43–50.

[7] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm," IEEE Trans. Knowl. Data Eng., vol. 24, no. 6, pp. 1146–1150, Jun. 2012.

[8] 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.