International Journal of Scientific Research in Computer Science, Engineering and Information Technology

© 2017 IJSRCSEIT | Volume 2 | Issue 5 | ISSN : 2456-3307

Measuring Software Quality Using Micro Interaction Metrics

A. Sreepradha

M.Tech Scholar, Computer Science and Engineering, JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh, India

ABSTRACT

Programming imperfection is a coding or logic mistakes that happen in application breakdown or wrong outcomes. Imperfection forecast is an imperative assignment in programming designing since Software quality is asset compelled movement, where it alludes to the constraint of staffing, gear, and different assets that are important to finish the undertaking, by anticipating programming inclined substances is to put the best work exertion on those elements. Engineer's cooperation's which are considered as one of the defects markers are caught in Mylyn, a shroud module with the assistance of smaller scale communication measurements (MIMs).

Keywords: Software quality, Mylyn, MIMs.

I. INTRODUCTION

Software quality can be related as the degree to which an approach, element, or process meets user or user desires. Only a defer in conveyance of programming it can cause genuine income loss. Sometimes demand for the software release to the market is a critical issue for companies in most sectors of the software market. In the meantime, despite the fact that is critical to meet such necessity (urgent demands) reckless quality commitment prompts responsibility and the fame in the market that is the reason quality commitment is the most fundamental thing before the product release. That's why software developers and software quality assurance team need an innovative technique that effectively predicts defects. That is how defect forecast has been engaging process in the research area in software engineering. Defects happen from human oversights, developers commit errors, therefore, abandons are infused, and programming comes up short. Some of the defect indicators are the complexity of source code, frequent code changes, previous defect information, code dependencies apart from these even the developer interactions like improper communication between developer and client, task switching, work interruptions etc are also considered as defect predict indicators. But the existing Source Code Metrics (CMs) and change History Metrics (HMs) are not enough to address the developer behavior. This detected correlations between behaviors of developers and quality of software production. It is preferable to make full use of developer's interaction. Behavior based software metrics can notify developers about quality practices for enhancing the high quality of software production. Designers need to comprehend what is happening to be a developed stage, get genuine certified if any rehashed blunder exists in their conduct when they are in chipping away at venture amid advancement. Here Micro Interaction Metrics (MIMs) are utilized. The task of those metrics is to seize the developer's behavioral interactions at some stage in the improvement manner. For this component to do, an Eclipse that is maximum broadly used Java IDE, in which complicated java initiatives are advanced. By making utilization of Mylyn, an Eclipse plug-in, for task context and storage recovery, it enables a developer to work effectively with many different tasks (such as bugs, problem reports or new features). Micro interaction metrics (MIMs) are interaction metrics used to hold the developer interaction information.

II. RELATED WORK

Past imperfections information are utilized to gauge future deformities by Kim et al. D'Ambros et al. conducted an extensive comparison of existing bug prediction approaches utilizing CMs, HMs, past defects, and the entropy of change metrics. Meneely et al. Proposed developer social network-based totally metrics to grasp the developer collaboration shape and expect defects using it. But all CMs, HMs doesn't deal with developer's interactions. Taek Lee, Han stated that programmer's interactions also affect the quality of software they proposed novel Micro Interaction Metrics (MIMs) that holds developers interaction data stored in the Mylyn data.

Multiple works such as bugs, problem reports or new functions can be done efficiently by developer with the help of Mylyn. Tasks are included into Mylyn. For all tasks which have been incorporated, Mylyn video display person activity and attempts to identify information regarding that particular task. Repositories namely Bugzilla, JIRA and Github etc are combined with Mylyn. Mylyn work is to record the developers situation and to store it, that records of the developers' situation such as editing or selecting files, can be reviewed when developer wants to review.

Micro Interaction Metrics (MIMs) are the interaction metrics to catch developer interactions that are related to committing errors. File level and task level are two classified design level of MIMs. The work of file-level MIMs is to catch specific interactions for a file in a project. *NumEditEvent* degree of MIMs for modifying interaction indicates quantity of selection activities for a selected file *NumSelectionEvent* file level metrics is for browsing interaction which indicates number of selection events observed for a particular file. *TimeSinceLastTask* is for time interval interaction which indicates time break since last task for a file. The task level MIMs represents properties per task. Some of the task level MIMs are *NumRareEdit* Number of edit events with low DOI attribute values, *NumParallelEdit* indicates number of files edited in parallel in a task session, *NumRepeatedEdit* number of files edited more than one time during a task session. *NumParallelBrws* for browsing interaction number of files browsed in parallel in a task session.

In defect prediction process, common bug categorization procedure has been used, which enables to forecast whether a given unknown file instance is buggy or not. In the first step all valid files are collected from Mylyn task session log as instances or datasets, instances that are presented in both, before and after the software release period are considered for extracting metrics and counting defects. Edited file information in Mylyn tasks are considered to count post defects as each task session log are directly attached with a bug report, it indicates whether the bug is fixed or not. In step two MIMs are extracted in file and task levels. For task sessions, unique interaction activities that target a report instance are aggregated from the metric extraction duration. File-level MIMs for the report example is computed with the specific event data. CMs and HMs are also extracted during metrics extraction period as to compare HMs, CMs, and MIMs. The Third step represents file instances, number of defects, number of developers and history of the files considered form the Mylyn task session. In fourth step bug prediction model is considered with the help of feature selection algorithm, in feature selection ten-fold cross- validation process had been used, the data sets are split into tenfold, the metrics that were taken in model construction. For classification purpose, classification algorithm is needed. The random forest pseudo code that executed in weka had been used. In fifth step evaluating the prediction models, F-measure had used, for predicting a buggy instances as clean or buggy.

III. PROPOSED SCHEME

To evaluate the software quality, comparison has been done in between naive bayes and correlation based feature subset algorithm from weka. Initially the datasets that contains CMs, HMs and MIMs contains metrics.
Some of the metrics that considered are *NumEditEvent*, *CntClassBase*, *AvgLineCmt* etc.

3.1 Load Dataset

Initially upload dataset filed allows users to upload files. Each dataset files are uploaded one after another. Considered CMs, HMs and MIMs measurements information records are uploaded.

3.2 Feature Selection Algorithms

Navie bayes and correlation based feature subset algorithms which are in weka have been considered. For each dataset file navies bayes and correlation feature subset algorithms are applied one after another. Confusion matrix is displayed which indicates buggy and clean for uploaded datasets by calculating F-measure.

3.3 Code Quality Graph

Here code quality graph is displayed by comparing navie bayes and correlation based feature subset algorithms. Graph is shown separately for every metric data file.

3.4 F-Measure Graph

To evaluate quality, F-measure graph is displayed by comparing Navie bayes and correlation based feature subset algorithms. Graph is displayed individually for each metric data file when uploaded one after another individually.

3.5 Code Quality of CMs, HMs and MIMs

It displays the code quality in numerical values of three metrics dataset files uploaded.

3.6 F-measure of CMs, HMs and MIMs

It displays F-measure values of all the three dataset files uploaded.

IV. EXPERIMENTAL RESULTS

In experimental results considered datasets like CMs, HMs and MIMs files are uploaded individually, each dataset metrics make use of navies bayes classifier first, it displays F-measure count and code quality count by classifying datasets as buggy and clean, again for the same data correlation based feature subset algorithm have applied it also displays F-measure count and code quality count by considering buggy and clean count. Comparison is done between navie bayes and correlation feature subset algorithm individually for each datasets files.

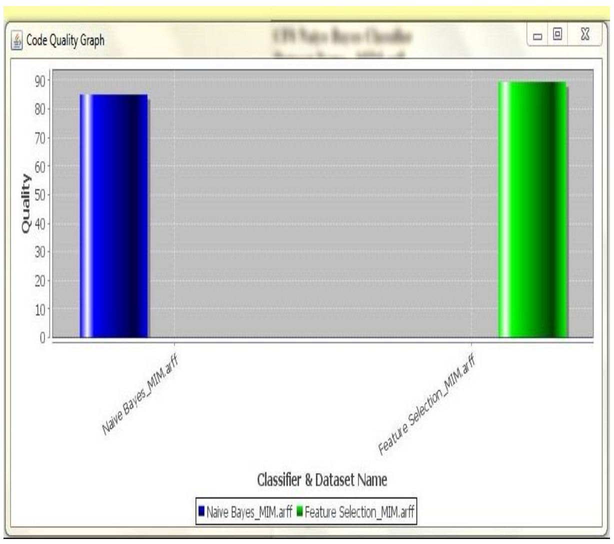
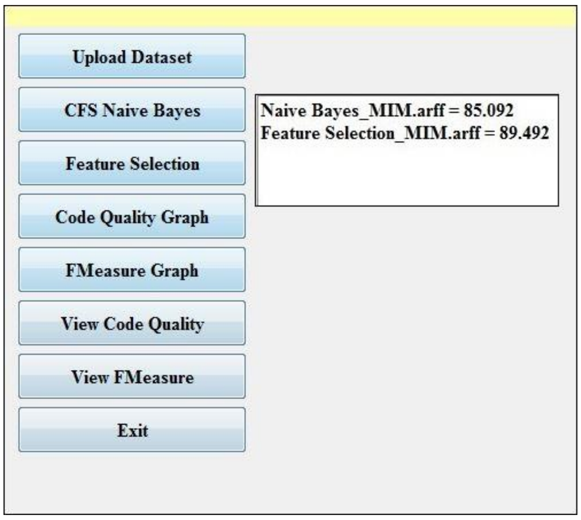


Figure 1. Code quality graph

Figure 1 code quality graph displays comparison between navie bayes and correlation feature subset algorithm for MIMs dataset file.
After code quality graph, F-measure graph is displayed, comparison between navie bayes and correlation feature subset algorithm.
F-measure count for naïve bayes and correlation feature subset can be seen. Same process is done for every dataset metrics (CMs, HMs, and MIMs).



```
Naive Bayes_MIM.arff = 85.092
Feature Selection_MIM.arff = 89.492
```

Figure 2. F-measure count

Figure 2 F-measure count indicates the improved instances count of correlation based feature subset algorithm when compared to naïve bayes. At last enhanced code quality and measure can be effortlessly recognisied.

V. CONCLUSIONS

By using Correlation feature subset algorithm code quality have been improved when compared to navie bayes algorithm. F-measure also improved when correlation feature subset algorithm is use. Code quality is improved from 85.9% to 89.4%.

VI. REFERENCES

[1]. Taek Lee, Jaechang Nam, Donggyun Han, Sunghun Kim, Member, IEEE, and Hoh Peter In "Developer Micro Interaction Metrics for Software Defect Prediction".

[2]. A. Hassan, "Predicting faults using the complexity of code changes," in Proc. 31st Int. Conf. Software Eng., 2009, pp. 78-88.

[3]. S. Kim, E. J. Whitehead Jr., and Y. Zhang, "Classifying software changes: Clean or buggy?" IEEE Trans. Softw. Eng., vol. 34, no. 2, pp. 181-196, Mar. 2008.

[4]. S. Kim, T. Zimmermann, E. J. Whitehead Jr., and A. Zeller, "Predicting faults from cached history," in Proc. 29th Int. Conf. Soft. Eng., 2007, pp. 489-498.

[5]. S. Kim, T. Zimmermann, E. J. Whitehead Jr., and A. Zeller, "Predicting faults from cached history," in Proc. 29th Int. Conf. Soft. Eng., 2007, pp. 489-498.

[6]. T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," IEEE Trans. Softw. Eng., vol. 33, pp. 2-13, Jan. 2007.

[7]. N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," in Proc. 28th Int. Conf. Softw. Eng., 2006, pp. 452-461.