Issue 7 volume 6 Nov.- Dec. 2017
ISSN 2249-9954

DESIGN AND ANALYSIS OF A NOVEL RANKING MODEL FOR MAPPING BUG REPORTS BASED ON FEATURE EVALUATION FOR THE RELEVANT FILES

KAMEPALLI LIKITHA [#1], **DR. B.PRAJNA**[#2]

[#1] M.Tech Scholar, Department of Computer Science and Technology,
Andhra University College of Engineering (A), Visakhapatnam, AP, India.

[#2] Professor, Department of Computer Science and Systems Engineering,
Andhra University College of Engineering (A), Visakhapatnam, AP, India.

ABSTRACT

As we all know that almost all software or mnc companies spend over 48 percent of total project cost in dealing with software bugs. This process is also known as software testing ,in which all software or product related bugs need to be identified and solved easily by assigning a correct developer for the existing or new bug. Till now the assignment of bugs for the developers is done in manual way, so there was a lot of maintenance cost and delay in solving the bugs. So in this proposed thesis we mainly designed and analyzed a new adaptive ranking approach that reduces the manual effort in assigning the bugs for the exact developer. Here in our proposed thesis, for each and every new bug, the ranking score of each source file is computed by considering all the features that are available in the bug as a weighted combination of features. Here the extracted bug features are trained automatically on previously solved bug reports that are available in the bug history database using a learning-to-rank technique. By using our proposed approach we mainly try to assign the bug for an accurate person for solving the bug and also find the rank or weight of bug by counting the number of attempts it got re-verified for solving that exact bug. Here as an extension we also maintained the history of bugs based on feature extraction for the appropriate keywords that was chosen by the admin for every bug report. By conducting various experiments on our proposed method, our simulation results clearly tell that our proposed method is very best in mapping the bugs to the relevant files based on the features that are extracted from the bugs.

Key Words: Software Bugs, Software Testing, Ranking Score, Feature Extraction, Bug Reports, Bug Maintenance.

I. INTRODUCTION

In the recent analysis which is conducted by a team of digital world survey, it is calculated and found that the quantity of knowledge/information within the digital world inflated from one hundred sixty two hexabytes in the year 2008 to 1048 hexabytes in the year 2017 [1]—about nineteen times the number of knowledge gift altogether the books ever written—and it continues to grow exponentially. This huge quantity of knowledge features a direct impact in pc knowledge scrutiny, which might be generally outlined because the discipline that mixes many components of knowledge and applied science to gather and analyze knowledge from pc systems in a very method that's admissible because the knowledge ought to have similarities between many collected data fields. In our explicit application domain, it always involves examining many thousands of files per pc. This activity exceeds the expert's ability of study and interpretation of knowledge. Therefore, ways for machine-controlled knowledge analysis, like those wide used for machine learning and data processing, are of predominate importance. Specially, algorithms for pattern recognition from the knowledge gift in text documents square measure promising, because it can hopefully become evident later within the paper[2],[3].

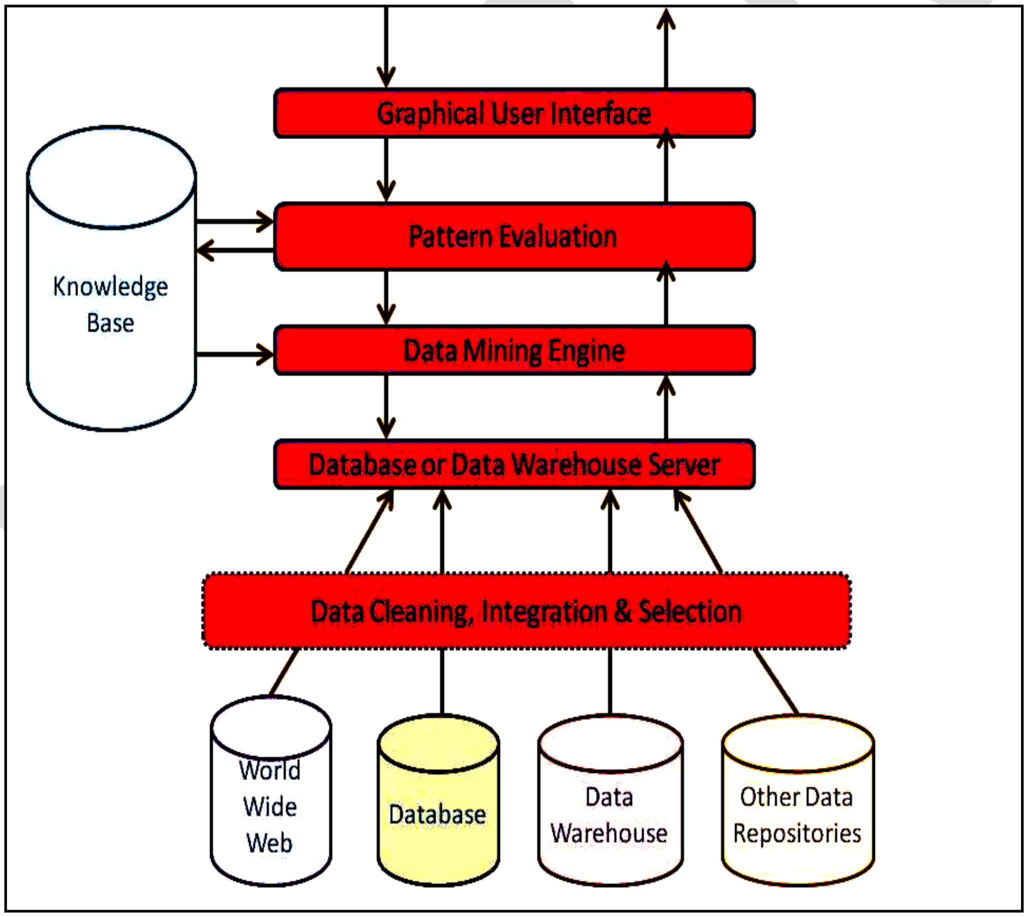


FIGURE 1. REPRESENTS THE TYPICAL ARCHITECTURE FOR THE PROCESS OF DATA MINING

From the above figure 1, we can clearly find out that for the process of data mining the data which should be taken as input will be collected from various resources like WWW, Database, Data

Warehouse and other data repositories. Once the input data is collected it will be then given to the process of data cleaning. Here in this data cleaning process the data will be cleaned and it will be processed in order to identify if there are any un-supervised data available in that input data. Once that data cleaning is completed now it will be processed for further iterations for data mining engine in order to process the knowledge base .Once the process of pattern evaluation is done then the data will be in turn converted into GUI.This graphical user interface is one which will give us the visualization of output. Here in our proposed thesis we are going to take a set of bugs that are arise while working on java technology, where each and every bug contains the bug description along with bug id and so on. Here the bugs should be easily identified with the help of features that are extracted based on the bug and that bugs should be properly assigned for a right developer in order to process the solution for that given bug.

In general mining software repositories is one form of domain, which mainly try to integrate the data mining domain with the software engineering problems and they will try to generate a solution for these at the end. In current software or IT development, each and every company following their own strategy for solving the bugs and for storing the large-scale databases on their own resources.e.g. Emails related to bugs, notification and specifications of old bugs or new bugs, sample code to test for the bugs and so on. All the primitive software analysis works are not completely suitable for solving the large-scale and complex data in software repositories. In that situation the data mining came into existence and evolved as a promising means to handle and solve all the software related bugs. A bug repository (a typical referred to as a software repository, in which all the details about the bugs and bugs information are stored inside it), plays an important role in managing software bugs. Almost all the software companies or MNC companies spend over 46 percent of their total project cost in fixing bugs ,which is arise in deployment phase or in maintenance phase.If we look into bug repository, a bug is monitored as a textual representation or description about the bug which is stored in the form of bug report. The most common platforms that were provided by the software companies are as follows: fault prediction, bug localization and reopened bug analysis .Of all the common platforms the two challenges that are related to the bug data is how effectively can a bug analyzed and how effectively it can be assigned for the correct developer based on feature extraction to solve the bug in exact time with no delay.

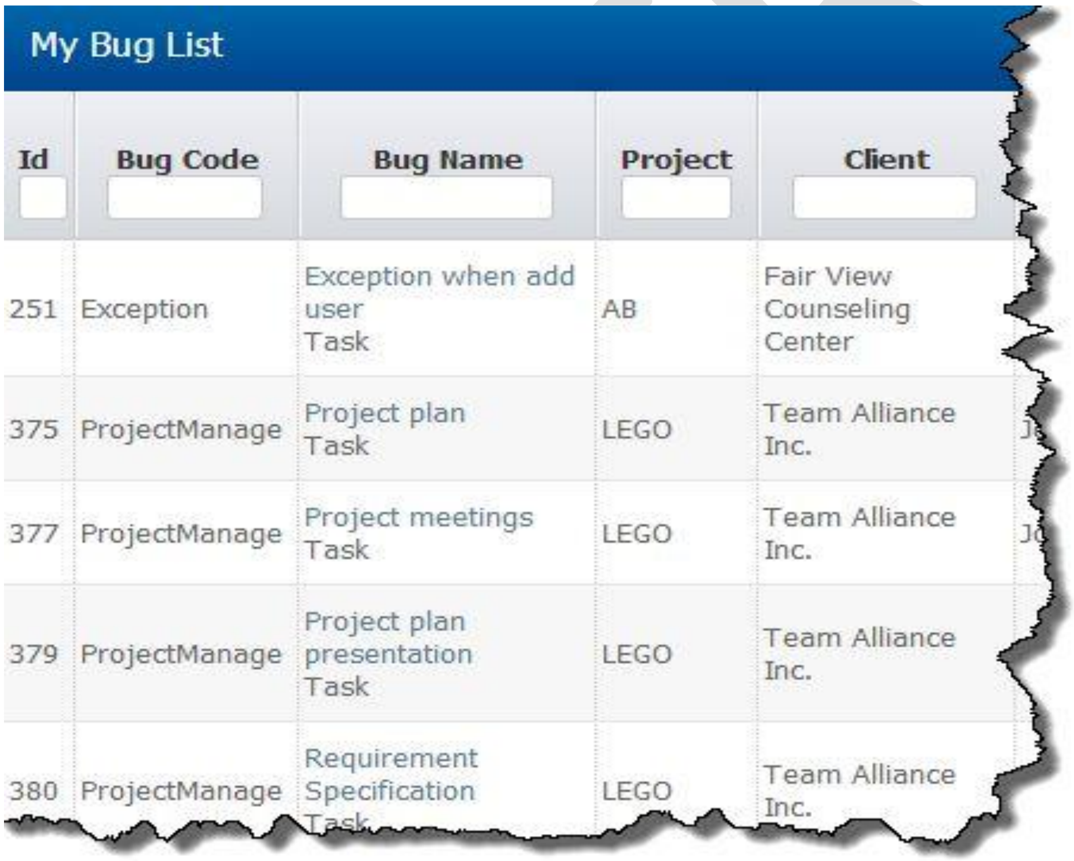
A developer who is assigned a bug and ask for rectification need to be in a abnormal behavior [4] and he need to be in a state to perform several code trails and conduct[5] a detailed code review report for finding the cause of that bug. In some situations the process becomes non trivial due to the cause like poor quality of bugs and diverse nature of the bug reports. In general almost all the bugs contain the missing information more than the useful information to sort out the bug report [6].A two well known authors like G.Bacch and D.Bired [7] conducted a survey for about 185 managers and 923 programmers who work on various IT companies and finally concluded that, for solving or rectifying the defects requires a very good level understanding of the code and very familiar knowledge about the relevant source code which to be mapped for the related bugs. While the number of source files in a project is usually large, the number of files that contain the information exactly related to the current bug report is usually very small. Therefore, we try to analyze and design an automatic approach that can extract the features from the bug automatically and try to map the bugs related to the exact developer in order to solve the bugs by testing from relevant source codes that are available in the bug repository.

II. RELATED WORK

In this section we mainly discuss about the related work that was carried out in finding the work that is related to proposed mapping bug reports based on instance or feature selection based on relevant files. Now let us discuss about this in detail:

ABOUT BUG REPOSITORY

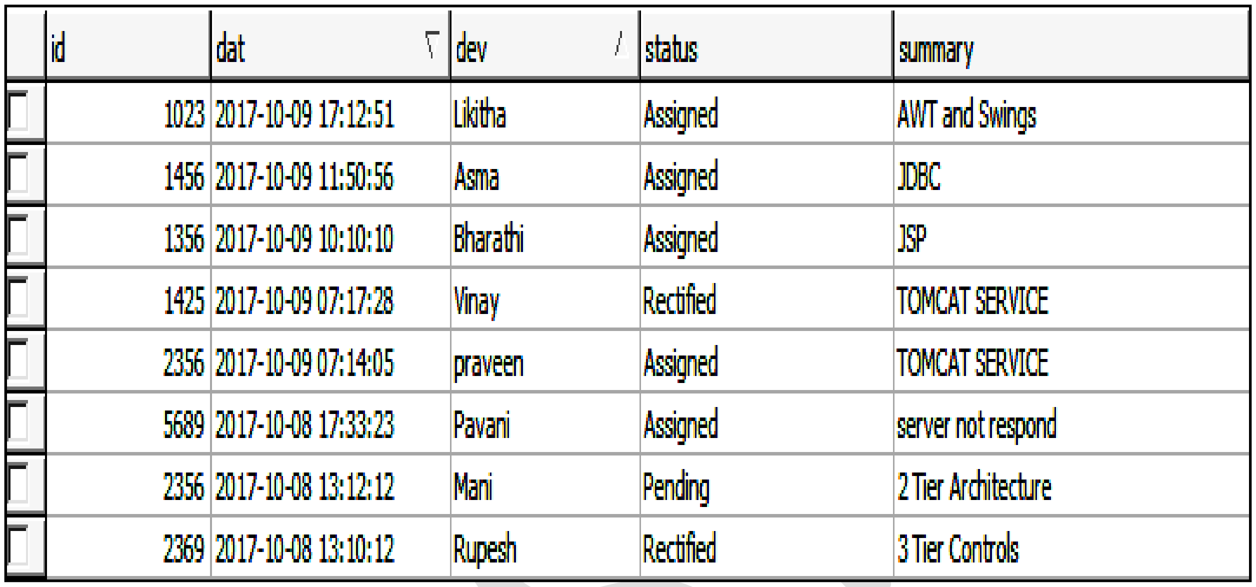
Bug repositories are one kind of storage medium that was mainly used for maintaining software bugs, e.g., A well known and an open source bug repository, Bugzilla [8]. Initially a bug is extracted and the features of bug is identified ,based on the features a reporter (typically a developer, a tester, or an end user)reads this bug description and insert this bug to the bug repository. A inserted bug is called as a bug report, which contain several fields for defining the information about the bug.


My Bug List

Id	Bug Code	Bug Name	Project	Client
251	Exception	Exception when add user Task	AB	Fair View Counseling Center
375	ProjectManage	Project plan Task	LEGO	Team Alliance Inc.
377	ProjectManage	Project meetings Task	LEGO	Team Alliance Inc.
379	ProjectManage	Project plan presentation Task	LEGO	Team Alliance Inc.
380	ProjectManage	Requirement Specification Task	LEGO	Team Alliance Inc.

FIGURE 2. REPRESENTS THE SAMPLE BUG REPOSITORY WITH A SET OF SOFTWARE BUGS POSTED BY VARIOUS CLIENTS

From the above figure.2, we can clearly see a bug repository contains many fields like bug-id, bug-code, bug-name, project name and client name. Here each and every filed has their own importance and we can clearly find out the description of each and every bug in a unique manner. Also there are some other items that are inserted in the bug report, which are not shown clearly in figure 2, but still have an importance in the bug report. Once a bug report is created, a human triager assigns this new or posted bug to a developer, who can fix the bug. This developer filed is identified as item assigned-to attribute in our bug report. This assigned-to field will change from one developer to another developer if the previously assigned developer cannot fix this bug within the specified time period. Hence our proposed application mainly deals with instance selection from the bug report and assigning a correct developer for the new bug automatically.



id	dat	dev	status	summary
1023	2017-10-09 17:12:51	Likitha	Assigned	AWT and Swings
1456	2017-10-09 11:50:56	Asma	Assigned	JDBC
1356	2017-10-09 10:10:10	Bharathi	Assigned	JSP
1425	2017-10-09 07:17:28	Vinay	Rectified	TOMCAT SERVICE
2356	2017-10-09 07:14:05	praveen	Assigned	TOMCAT SERVICE
5689	2017-10-08 17:33:23	Pavani	Assigned	server not respond
2356	2017-10-08 13:12:12	Mani	Pending	2 Tier Architecture
2369	2017-10-08 13:10:12	Rupesh	Rectified	3 Tier Controls

FIGURE 3. REPRESENTS THE BUG HISTORY TABLE FOR THE POSTED BUGS

From the above figure 3,we can clearly identify that a developer, who is assigned to a new bug report, try to solve the assigned bug from the date of assignment with the previous knowledge what he have on that bug,[9], [10]. Initially, the developer try to concentrate more time and put all his effort to understand the bug report and then try to examine the history of the bug, whether the similar bug is already solved or whether this bug is the first time to solve. An item status of a bug report is changed according to the current result of handling this bug until the bug is completely fixed. All the status changes will be updated in the status field which is clearly shown in the figure 1.Initailly the bug will be in Assigned status and once the bug is still in process of understanding by the developer then the bug status will be Pending and if the bug was successfully solved and solution is reported for the client, then the bug status will be changed to Rectified. If we continue the same assignment process with manual bug triage by a human triager, it is s time-consuming and error-prone since the number of daily bugs is large to correctly assign and a human triager is hard to master the knowledge about all the bugs [12]. Existing work employs the approaches based on text classification to assist bug triage, e.g., [11]. In such approaches, the summary and the description of a bug report are extracted as the textual content while the developer who can fix this bug is marked as the label for classification. Then techniques on text classification can be used to predict the developer for a new bug. In details, existing bug reports with their developers are formed as a training set to train a classifier (e.g., Naive Bayes, a typical classifier in bug triage [13]); new bug reports are treated as a test set to examine the results of the classification. In Fig. 2a, we illustrate the basic framework of bug triage based on text classification.

III. THE PROPOSED LEARNING-TO-RANK TECHNIQUE (OR) DATA REDUCTION BASED ON INSTANCE/FEATURE SELECTION

In this section we mainly discuss about the proposed **DR-IS/FS Algorithm** for data reduction based on instance or feature selection. Now let us discuss about this proposed **DR-IS/FS Algorithm** in detail as follows:

PRELIMINARY KNOWLEDGE

In this section, we first present how to apply instance selection and feature selection to bug data, i.e., data reduction for bug triage. Then, we list the benefit of the data reduction. In this section, we mainly observe how a bug report is mapped to a document and a related developer is mapped to the bug based on the features that are matched with that developer domain. Here each and every new bug or existing bug, is converted into a problem of text classification and is automatically solved with mature text classification technique. To improve the accuracy of text classification techniques for mapping the bus to relevant files, some further techniques are investigated, e.g., an adaptive filtering approach. In this proposed thesis, we address the problem of data reduction for bug mapping, i.e., how to reduce the bug data to save the labor cost of developers and improve the quality to facilitate the process of bug triage.

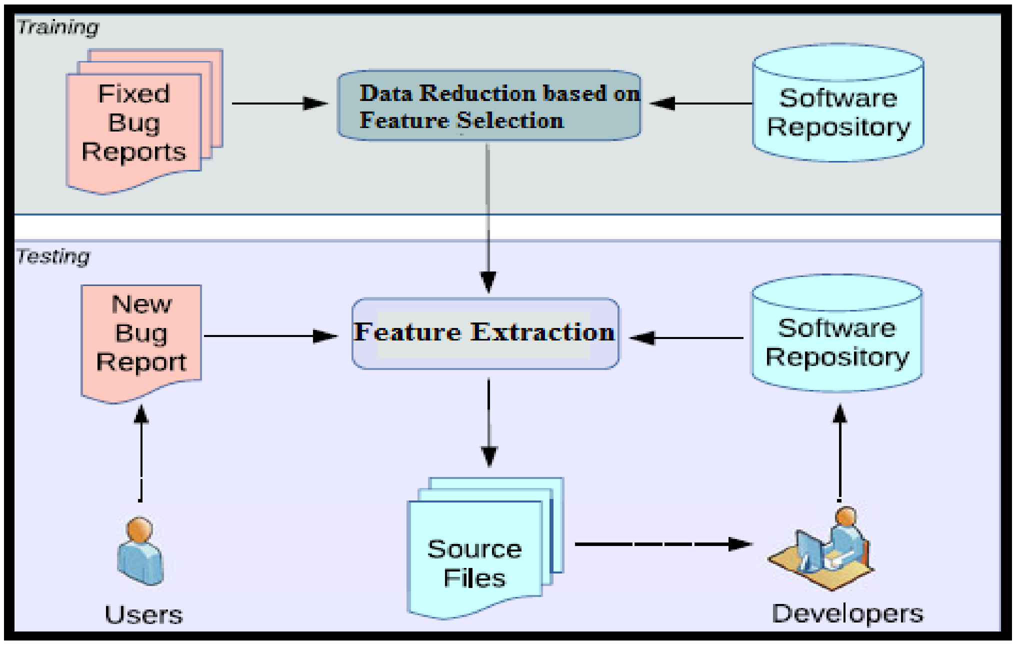


FIGURE 4. REPRESENTS THE PROPOSED DR-IS/FS ALGORITHM FOR MAPPING BUG REPORTS BASED ON FEATURE EVALUATION FOR THE RELEVANT FILES

From the above figure 4,we can clearly find out the proposed DR-IS/FS algorithm ,where the architecture contains mainly two roles: One is user role and other is Developer role. Here the users are nothing but may be a customer or a client who got a bug and try to post the new bug into the bug repository. Once the user like client or customer post the new bug, the system should automatically identify the features that are available in the posted bug report and for this features extraction we use the FS algorithm. Once the features are extracted from bug report, then that bug will be assigned to the developer with the relevant source files to solve the bug easily and exactly by the concern developer who is expertise for that appropriate feature. Now the developer will receive the bug request assigned by the system and now the developer will initially check the bug features and he will select some instance features as search keywords and try to search whether same featured bugs are available in the bug repository. If the developer finds the bug matched with any of the bug that is available in the bug repository, then this bug will be undergone for the training state. Here in this phase the developer will come to know clearly about the bug status and he will give conclusion for that bug report and finally he will store the status of the bug in Fixed Bug Reports table.

ALGORITHM: DATA REDUCTION BASED ON FS-IS

INPUT:

Training set denoted with **T** with **n** words and **m** bug reports,

reduction order FS ----> IS

final number nF of words,

final number mI of bug reports,

OUTPUT: reduced data set T_{FI} for bug triage

1) Apply FS to **n** words of **T** and calculate objective values for all the words.

2) Select the top n_F words of T and generate a training set T F

3) Apply IS to mI bug reports of T F

3) Terminate IS when the number of bug reports is equal to or less than m_I and generate the final training set T_{FI} .

In our proposed application, a bug data set is converted into a text matrix with two dimensions, namely the bug dimension and the word dimension. In our work, we leverage the combination of instance selection and feature selection to generate a reduced bug data set. We replace the original data set with the reduced data set for bug solution. 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) [14] while feature selection aims to obtain a subset of relevant features (i.e., words in bug data) [15]. In our work, we employ the combination of instance selection and feature selection. To distinguish the orders of applying instance selection and feature selection, we give the following denotation. Given an instance selection algorithm IS and a feature selection algorithm FS, we use FS → IS to denote the bug data reduction, which first applies FS and then IS; on the other hand, IS → FS denotes first applying IS and then FS.

IV. . IMPLEMENTATION AND ITS METHODOLOGY

Implementation is the stage where the theoretical design is converted into programmatically manner. In this stage we will divide the application into a number of modules and then coded for deployment. We have implemented the proposed concept on Java programming language with JEE as the chosen language in order to show the performance this proposed application. The front end of the application takes JSP,HTML and Java Beans and as a Back-End Data base we took My SQL data base along with a Some predefined bug repository which contain a set of bugs related to Java Programming language. The application is divided mainly into following 3 categories and in each and every category again there are several modules. Now let us discuss about them in detail as follows:

1. Manager Module
2. Team Leader Module
3. Developer Module

1. MANAGER MODULE

In this module, the Admin has to login by using valid user name and password. After admin login successful he can do some operations such as add projects, recruit developer, view all bugs status, list all projects, list all assigned projects, list all users, view searched history.Here the manager is one who is totally responsible for monitoring all the bugs that are posted by various clients either in the phase of development or in the maintenance stage.

2. TEAM LEADER MODULE

In this module, the TL has to login by using valid user name and password. After login successful he can do some operations such as view the list of bugs that are assigned to him from manager. He can assign the list of bugs based on feature or instance selection. Here the bugs are categorized based on instances that are available on the bug report.Here the team leader can see the status of all bugs that are posted by the clients. He can re-assign the bugs which are in not assigned status or not rectified status.

3. DEVELOPER MODULE

In this module, initially the developer will be recruited by the manager, so once he got registered the developer has to login by using valid user name and password. After login successful he can do some operations such as view the list of bugs that are assigned to him from TL. He can give solution for that bug report after verifying the source code that is available with him for that relevant bug. Once the bug was solved by the developer the bug status will be automatically converted into rectified status and this will be in-turn posted to manager and TL.So once if the bug was rectified the bug will be automatically disabled from the bug list table of that developer.

V. CONCLUSION

In this proposed thesis, we for the first time developed a novel method for identifying a bug, developers use not only the content of the bug report but also domain knowledge relevant to the software project. We introduced a learning-to-rank approach that emulates the bug finding process employed by developers. Here we took some sample bugs in java platform and we tested those bugs for various bug reports based on the feature extraction.Our proposed method initially take all the features into account and then map those features with set of already solved bug reports and if there was any bugs that are matched with this current features,they will be extracted and send to the developer for that software testing. By conducting various experiments on our proposed framework, we finally came to an conclusion that our proposed approach is robust and easy to solve the bugs automatically with the help of instance or feature selection.

VI. REFERENCES

[1] G. Salton and C. Buckley, “Term weighting approaches in automatic text retrieval,” *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.

[2] A. Strehl and J. Ghosh, “Cluster ensembles: A knowledge reuse framework for combining multiple partitions,” *J. Mach. Learning Res.*, vol. 3, pp. 583–617, 2002.

[3] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, "Data Preprocessing for Supervised Learning", *International Journal of Computer Science*, 2006, Vol 1 N. 2, pp 111–117.

[4] T. D. LaToza and B. A. Myers, “Hard-to-answer questions about code,” in Proc. Eval. Usability Programm. Lang. Tools, New York,NY, USA, 2010, pp. 8:1–8:6.

[5] T. Lee, J. Nam, D. Han, S. Kim, and H. P. In, “Micro interaction metrics for defect prediction,” in Proc. 19th ACM SIGSOFT Symp.13th Eur. Conf. Found. Softw. Eng., New York, NY, USA, 2011,pp. 311–321.

[6] 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, New York, NY, USA, 2010, pp. 301–310.

[7] A. Bacchelli and C. Bird, “Expectations, outcomes, and challenges of modern code review,” in Proc. Int. Conf. Softw. Eng., Piscataway, NJ, USA, 2013, pp. 712–721.

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

[9] E. Murphy-Hill, T. Zimmermann, C. Bird, and N. Nagappan, “The design of bug fixes,” in Proc. Int. Conf. Softw. Eng., 2013, pp. 332– 341.

[10] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, € and C. Weiss, “What makes a good bug report?” IEEE Trans. Softw. Eng., vol. 36, no. 5, pp. 618–643, Oct. 2010

[11] V. Cerveron and F. J. Ferri, “Another move toward the minimum consistent subset: A tabu search approach to the condensed nearest neighbor rule,” IEEE Trans. Syst., Man, Cybern., Part B, Cybern., vol. 31, no. 3, pp. 408–413, Jun. 2001.

[12] D. Cubrani c and G. C. Murphy, “Automatic bug triage using text categorization,” in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng., Jun. 2004, pp. 92–97.

[13] Eclipse. (2014). [Online]. Available: http://eclipse.org/

[14] 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.

[15] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” J. Mach. Learn. Res., vol. 3, pp. 1157–1182, 2003

VII. ABOUT THE AUTHORS

KAMEPALLI LIKITHA is currently pursuing her 2 Years M.Tech in Department of Computer Science and Technology at Andhra University College of Engineering, Visakhapatnam, Andhra Pradesh, India. Her area of interests includes Data Mining.

DR. B.PRAJNA is currently working as a Professor in Department of Computer Science and Systems Engineering at Andhra University College of Engineering, Visakhapatnam, Andhra Pradesh, India. She has more than 15 years of teaching experience. Her research interest includes Data Mining.