International Journal of Science Engineering and Advance Technology

# Efficient Ranking and Clustering Approach for Bug Fixing In Software Projects

[1]Durga Doddipatla, [2]Ch. Subhash
[1,2]Department of Computer Science & Engineering, Kakinada Institute of Engineering &Technology,Korangi.

**ABSTRACT:**
A bug report gives data that could help in settling a bug, with the general point of enhancing the product quality. A substantial number of bug reports could be opened amid the advancement life-cycle of a product product.Given a bug report, the positioning score of each source document is registered as a weighted mix of a variety of highlights, where the weights are prepared consequently on already fathomed bug reports utilizing a figuring out how to-rank procedure.

**KEYWORDS:** ranking model, bug reports, mapping

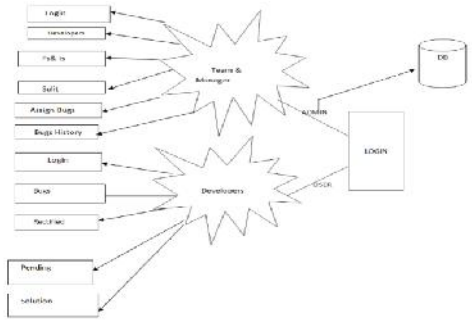
**INTRODUCTION:**
An engineer who is assigned a bug report more often than not needs to imitate the strange conduct and perform code surveys [4] keeping in mind the end goal to discover the reason. Be that as it may, the assorted variety and uneven nature of bug reports can make this procedure nontrivial. Fundamental data is regularly absent from a bug report [11]. Bacchelli and Bird [4] reviewed 165 directors and 873 software engineers, and detailed that discovering abandons requires an abnormal state comprehension of the code and commonality with the applicable source code documents. In the study, 798 respondents addressed that it requires investment to audit new records. While the quantity of source documents in an undertaking is generally extensive, the quantity of records that contain the bug is typically little. Therefore, we trust that a programmed approach that positioned the source records concerning their pertinence for the bug report could accelerate the bug discovering process by narrowing the inquiry to fewer perhaps new documents.

**LITERATURE SURVEY:**
[1] Present an approach to feature identification and comparison for large object-oriented multi-threaded programs using both static and dynamic data. We use processor emulation, knowledge filtering, and probabilistic ranking to overcome the difficulties of collecting dynamic data
It doesn't identifies the bug fixes source code files and unable to display the ranked source files.
[2] Propose new statistical analyses of static and dynamic data to accurately identify features in large multithreaded object-oriented programs. We draw inspiration from epidemiology to improve previous approaches to feature identification and develop an epidemiological metaphor
It doesn't identifies the bug fixes source code files and unable to display the ranked source files and cannot consider the all features.

**PROBLEM DEFINITION:**
The approach is evaluated on a fixed version of the source code package of every project, which is problematic due to potential contamination with future bug-fixing information.
one-phase model uses only previously fixed files as labels in the training process, and therefore cannot be used to recommend files that have not been fixed before when being presented with a new bug report.

**PROPOSED APPROACH:**
The proposed ranking model in conjunction with a learning-to-rank technique; using the file dependency graph to define features that capture a measure of code complexity.
Algorithm used: ranking model and hierarchal clustering

**SYSTEM ARCHTECTURE:**



**PROPOSED METHODOLOGY:**

**SYSTEM CONSTRUCTION:**

An approach that leverages project knowledge through functional decomposition of source code, API descriptions of library components, the bug-fixing history, the code change history.

**RANKING FUNCTION:**

The ranking function is defined as a weighted combination of features, measure relevant relationships between the bug report and the source code file. While a bug report may share textual tokens with its relevant source files.

**FEATURE REPRESENTATION:**

Abugreport source file pair be represented as a vector of k features.

These are features that depend on both the bug report r and the source code file s.

A query dependent feature represents a specific relationship between the bug report and the source file.

It may be useful in determining directly whether the source code file s contains a bug that is relevant for the bug report.

**COLLABORATIVE FILTERING SCORE:**

It has been observed in that a file that has been fixed before may be responsible for similar bugs.This collaborative filtering effect has been used before in other domains to improve the accuracy of recommender systems, consequently it is expected to be beneficial in our retrieval setting, too. Given a bug report r and a source code file s, let br(r , s) be the set of bug reports for which file s was fixed before r was reported. The feature computes the textual similarity between the text of the current bug report r and the summaries of all the bug reports in br(r , s). This feature is query-dependent.

**PROPOSED ALGORITHM:**

**LEARNING-TO-RANK APPROACH TECHNIQUE:**

Input:r,s

Output:relevant source code file.

STEP1:mappingsourcecode files and bug reports as a feature vector .

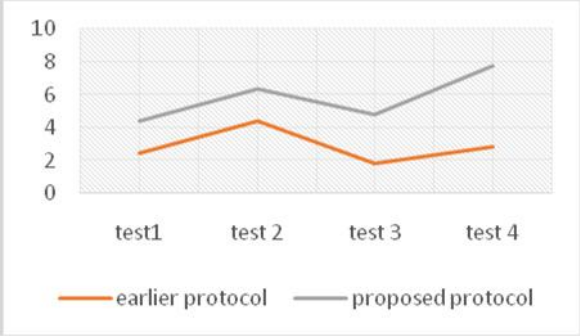
STEP2: the query and the source code document are represented as vectors of term weights.

STEP3:calculation of frequency factor in the overall term weight.

STEP4:grouping the sourcode documents relevant to bug reports by using hierarchal clustering from repository.

STEP5:displaying relevant source code files based on given bug report .

**RESULTS:**




The proposed approach shows productive execution as far as security and correspondence and calculation overhead contrasted with before technique.

**CONCLUSON:**

The proposed Ranking function along with clustering approach decreases the time to bug fixing delays for developers.

In future adopt hadoop framework along with classification model to improve the mapping accuracy for source code documents along with bug reports.

**EXTENSION WORK:**

Hierarchal clustering algorithm is used to cluster the bug reports in hierarchal manner based on projects.

**REFERENCES:**

[1] G. Antoniol and Y.-G. Gueheneuc, “Feature identification: A novel approach and a case study,” in Proc. 21st IEEE Int. Conf. Softw. Maintenance,Washington, DC, USA, 2005, pp. 357–366.

[2] G. Antoniol and Y.-G. Gueheneuc, “Feature identification: An epidemiological metaphor,” IEEE Trans. Softw. Eng., vol. 32, no. 9, pp. 627–641, Sep. 2006.

[3] B. Ashok, J. Joy, H. Liang, S. K. Rajamani, G. Srinivasa, and V. Vangala, “Debugadvisor: A recommender system for debugging,” in Proc. 7th Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng., New York, NY, USA, 2009, pp. 373–382.

[4] 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.

[5] S. K. Bajracharya, J. Ossher, and C. V. Lopes, “Leveraging usage similarity for effective retrieval of examples in code repositories,” in Proc. 18th ACM SIGSOFT Int. Symp. Found. Softw. Eng., New York, NY, USA, 2010 pp. 157–166.

[6] R. M. Bell, T. J. Ostrand, and E. J. Weyuker, “Looking for bugs in all the right places,” in Proc. Int. Symp. Softw. Testing Anal., New York, NY, USA, 2006, pp. 61–72.

[7] N. Bettenburg, S. Just, A. Schr€oter, C. Weiss, R. Premraj, and T. Zimmermann, “What makes a good bug report?” in Proc. 16th ACM SIGSOFT Int. Symp. Found. Softw. Eng., New York, NY, USA, 2008, pp. 308–318.

[8] T. J. Biggerstaff, B. G. Mitbander, and D. Webster, “The concept assignment problem in program understanding,” in Proc. 15th Int. Conf. Softw. Eng., Los Alamitos, CA, USA, 1993, pp. 482–498.

[9] D. Binkley and D. Lawrie, “Learning to rank improves IR in SE,” in Proc. IEEE Int. Conf. Softw. Maintenance Evol., Washington, DC, USA, 2014, pp. 441–445.

[10] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” J. Mach. Learn. Res., vol. 3, pp. 993–1022 Mar. 2003.

[11] 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.

[12] B. Bruegge and A. H. Dutoit, Object-Oriented Software Engineering Using UML, Patterns, and Java, 3rd ed. Upper Saddle River, NJ, USA, Prentice-Hall, 2009.

[13] Y. Brun and M. D. Ernst, “Finding latent code errors via machine learning over program executions,” in Proc. 26th Int. Conf. Softw. Eng.,Washington, DC, USA, 2004, pp. 480–490.

[14] M. Burger and A. Zeller, “Minimizing reproduction of software failures,” in Proc. Int. Symp. Softw. Testing Anal., New York, NY, USA, 2011 pp. 221–231.

[15] R. P. L. Buse and T. Zimmermann, “Information needs for software development analytics,” in Proc. Int. Conf. Softw. Eng., Piscataway, NJ, USA, 2012, pp. 987–996.

**DurgaDoddipatla**is a student of Kakinada Institute of Engineering & Technology, Korangi. Currently, she is pursuing M.Tech specializing in SE department. She awarded B.Tech specialized in CSE from Kakinada Institute of Engineering & Technology, Korangi

**Mr.Ch.Subhash,**M.Tech,M.B.A is working as an Assistant Professor, Department of Computer Science and Engineering, at Kakinada Institute of Engineering and Technology,korangi.