International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

IMPROVING AND AUTOMATED BUG PREDECTION SYSTEM USING MACHINE LEARNING ALGORITHMS

[1]K.JAGADHISH, [2] Dr.B.HARI BABU

[1]M.TECH DEPT OF CSE, KAKINADA INSTITUTE OF TECHNOLOGICAL SCIENCES, RAMACHANDRAPURAM, ANDHRAPRADESH, INDIA, 533255

[2]ASSISTANT PROFESSOR, KAKINADA INSTITUTE OF TECHNOLOGICAL SCIENCES, RAMACHANDRAPURAM, ANDHRAPRADESH, INDIA, 533255

ABSTRACT

Several techniques have been proposed to accurately predict software defects. These techniques generally exploit characteristics of the code artefacts (e.g., size, complexity, etc.) and/or of the process adopted during their development and maintenance (e.g., the number of developers working on a component) to spot out components likely containing bugs. While these bug prediction models achieve good levels of accuracy, they mostly ignore the major role played by human-related factors in the introduction of bugs. Previous studies have demonstrated that focused developers are less prone to introduce defects than non-focused developers. According to this observation, software components changed by focused developers should also be less error prone than components changed by less focused developers. We capture this observation by measuring the scattering of changes performed by developers working on a component and use this information to build a bug prediction model. Such a model has been evaluated on 26 systems and compared with four competitive techniques. The achieved results show the superiority of our model, and its high complementarily with respect to predictors commonly used in the literature. Based on this result, we also show the results of a "hybrid" prediction model combining our predictors with the existing ones.

1. INTRODUCTION

Bug prediction techniques are used to identify areas of software systems that are more likely to contain bugs. These prediction models represent an important aid when the resources available for testing are scarce, since they can indicate where to invest such resources. The scientific community has developed several bug prediction models that can be roughly classified into two families, based on the information they exploit to discriminate between "buggy" and "clean" code components. The first set of techniques exploits product metrics (i.e., metrics capturing intrinsic characteristics of the code components, like their size and complexity) [1], [2], [3], [4], [5], while the second one focuses on process metrics (i.e., metrics capturing specific aspects of the development process, like the frequency of changes performed to code components) [6], [7], [8], [9], [10], [11], [12]. While some studies highlighted the superiority of these

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

latter with respect to the product metric based techniques [7], [13], [11] there is a general consensus on the fact that no technique is the best in all contexts [14], [15]. For this reason, the research community is still spending effort in investigating under which circumstances and during which coding activities developers tend to introduce bugs (see e.g., [16], [17], [18], [19], [20], [21], [22]). Some of these studies have highlighted the central role played by developer-related factors in the introduction of bugs.In particular, Eyolfson et al. [17] showed that more experienced developers tend to introduce less faults in software systems. Rahman and Devanbu [18] partly contradicted the study by Eyolfson et al. by showing that the experience of a developer has no clear link with the bug introduction. Bird et al. [20] found that high levels of ownership are associated with fewer bugs. Finally, Posnett et al. [22] showed that focused developers (i.e., developers focusing their attention on a specific part of the system) introduce fewer bugs than unfocused developers. Although such studies showed the potential of human-related factor sin bug prediction,thisinformation is not captured in state-of-the-art bug prediction models based on process metrics extracted from version history. Indeed, previous bug prediction models exploit predictors based on (i) the number of developers working on a code component [9] [10]; (ii) the analysis of changeproneness [13] [11] [12]; and (iii) the entropy of changes [8]. Thus, despite the previously discussed finding by Posnett et al. [22], none of the proposed bug prediction models considers how focused the developers performing changes are and how scattered these changes are. In our previous work [23] we studied the role played by scattered changes in bug prediction. We defined two measures, namely the developer's structural and semantic scattering. The first assesses how "structurally far" in the software project the code components modified by a developer in a given time period are.The "structural distance" between two code components is measured as the number of subsystems one needs to cross in order to reach one component from the other. The second measure (i.e., the semantic scattering) is instead meant to capture how much spread in terms of implemented responsibilities the code components modified by a developer in a given time period are. The conjecture behind the proposed metrics is that high levels of structural and semantic scattering make the developer more error-prone. To verify this conjecture, we built two predictors exploiting the proposed measures, and we used them in a bug prediction model. The results achieved on five software systems showed the superiority of our model with respect to (i) the Basic Code Change Model (BCCM) built using the entropy of changes [8] and (ii) a model using the number of developers working on a code component as predictor [9] [10]. Most importantly, the two scattering measures showed a high degree of complementarity with the measures exploited by the baseline prediction models. In this paper, we extend

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

our previous work [23] to further investigate the role played by scattered changes in bug prediction. In particular we: 1) Extend the empirical evaluation of our bug prediction model by considering a set of 26 systems. 2) Compare our model with two additional competitive approaches, i.e., a prediction model based on the focus metrics proposed by Posnett et al. [22] and a prediction model based on structural code metrics [24], that together with the previously considered models, i.e., the BCCM proposed by Hassan [8] and the one proposed by Ostrand et al. [9] [10], lead to a total of four different baselines considered in our study. 3) Devise and discuss the results of a hybrid bug prediction model, based on the best combination of predictors exploited by the five prediction models experimented in the paper. 4) Provide a comprehensive replication package [25] including all the raw data and working data sets of our studies. The achieved results confirm the superiority of our model, achieving a F-Measure 10.3% higher, on average, than the change entropy model [8], 53.7% higher, on average, with respect to what achieved by exploiting the number of developers working on a code component as predictor [9], 13.3% higher, on average, than the FMeasure obtained by using the developers' focus metric by Posnett et al. [22] as predictor, and 29.3% higher, on average, with respect to the prediction model built on top of product metrics [1]. The two scattering measures confirmed their complementarity with the metrics used by the alternative prediction models. Thus, we devised a "hybrid"model providing an average boost inprediction accuracy (i.e., F-Measure) of +5% with respect to the best performing model (i.e., the one proposed in this paper).

II. EXISTING SYSTEM

To aid in finding appropriate developers, automatic bug triaging approaches have been proposed in the existing. Many of these approaches use the vector space model (VSM) to represent a bug report, i.e., a bug report is treated as a vector of terms (words) and their counts. However, developers often use various terms to express the same meaning. The same term can also carry different meanings depending on the context. These synonymous and polysemous words cannot be captured by VSM.

Various topic modeling algorithms are proposed in the literature including Latent Semantic Indexing/Analysis (LSA), probabilistic LSA (pLSA), and Latent Dirichlet Allocation (LDA). Among the three, LDA is the most recently proposed and it addresses the limitations of LSA and pLSA.

DISADVANTAGES OF EXISTING SYSTEM:

LDA considers a document as a random mixture of latent topics, where a topic is a random mixture of terms. One or few features can be only taken into consideration. Lower accuracy. More complex, More time taken

III. PROPOSED SYSTEM:

We extend LDA and propose a new topic model named multi-feature topic model (MTM) for the bug triaging problem. Since

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

a bug report has multiple features (e.g., product affected by the bug, component affected by the bug, etc.), MTM considers the features of a bug report when it converts terms in the textual description of the report (i.e., texts in the summary and description fields of the report) to their corresponding topics in the topic space. Given a bug report with a particular feature combination (i.e., product component combination), MTM converts a word in the bug report, to a topic.

We refer to a feature as a categorical field in a bug report that a bug reporter can fill when the reporter submits a bug report. These fields include the product, component, reporter, priority, severity, OS, version, and platform fields. We exclude the natural language descriptions in the bug reports, which includes the contents of the summary and description fields, as the features since they are not categorical in nature.

In this paper, we use the product-component combination as the input feature combination, since product and component are two of the most important features that describe a bug. Given a bug report with a particular feature combination, MTM converts a term in the bug report to a topic by putting special emphasis on the appearances of the word in bug reports with the same feature combination, without ignoring the word appearances in all other bug reports.

ADVANTAGES OF PROPOSED SYSTEM:

MTM considers each combination of features as a random mixture of latent topics, where a topic is a random mixture of terms. MTM is an extensible topic model, where one or more features can be taken into consideration. We propose a new approach for bug triaging which leverages MTM. We take as input a training set of bug reports (whose fixers are known) and a new bug report whose fixer is to be predicted. Our approach, named TopicMiner MTM computes the affinity of a developer to a new bug report, based on the reports that the developer fixed before. To do this, we compare the topics that appear in the new bug report with those in the old reports that the developer has fixed before

IV. MODULES

4.1 Admin

In this module, the Admin has to login by using valid user name and password. After login successful he can do some operations such as View all Project Developers and Authorize, View all Managers and Authorize, View all Team Members based on project, View all Bugs details from team members and manager and given solution with req date and res date , View number of time occures same Bug for a project and give link to show in Chart, View No.Of team members for each project assigned in Chart.

4.1.1 View and Authorize Users

In this module, the admin can view the list of users who all registered. In this, the admin can view the user's details such as, user name, email, address and admin authorizes the users.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

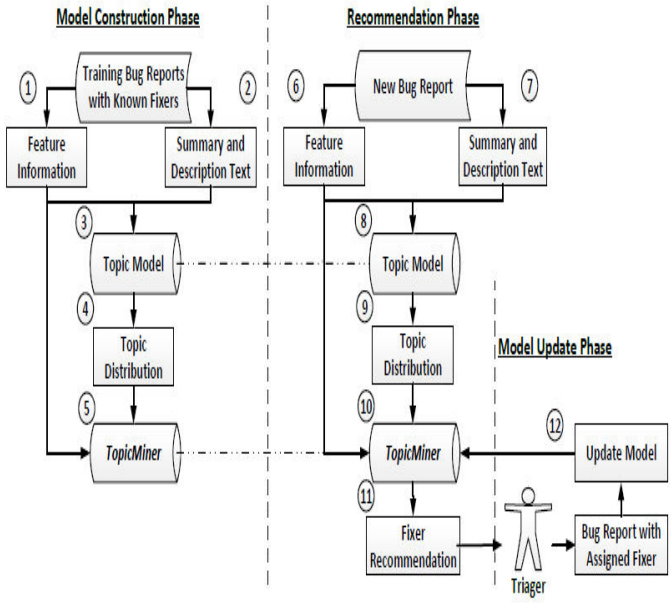
4.2 Manager

In this module, there are n numbers of managers are present. Manager should register before doing any operations. Once manager registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful manager will do some operations like View all Team Members based on project, Add Projects with Project name and start date and end date,expected date, View all employees and select emp to Add Project, View all added project details and give edit option(proj desc,proj module name,Add Proj Sub modules), View all team members defects and give solution or allote to other team members, Add extension dates for the project to deliver, View all projects status from team members, View all complexity of the project while developing.

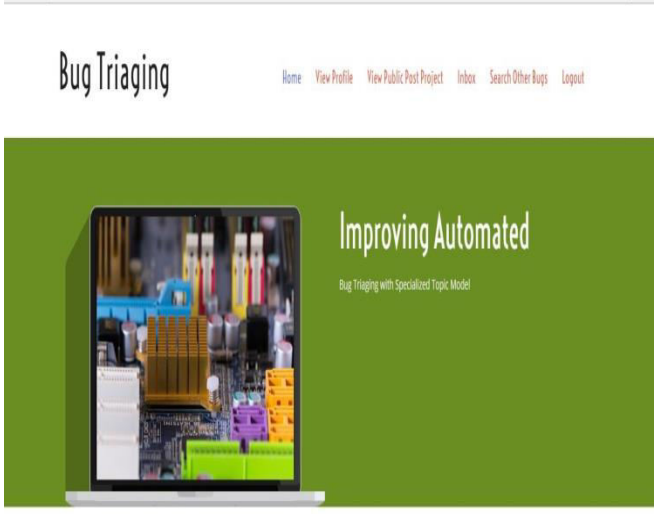
4.3 Project Developers

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like View Your Profile with company Name,View all Assigned Projects with all details,Set defects and send to corresponding team member,Set defects and send to corresponding team manager,view all solutions based on team member and team manager,View all projects and select your projects to show the status like Open,closed,Completed,Withdrawn ,View all team members defects and give solution, View all assigned Projects extension details, add complexity of the project while developing.

V. SYSTEM ARCHITECTURE



VI . SCREEN SHOTOS



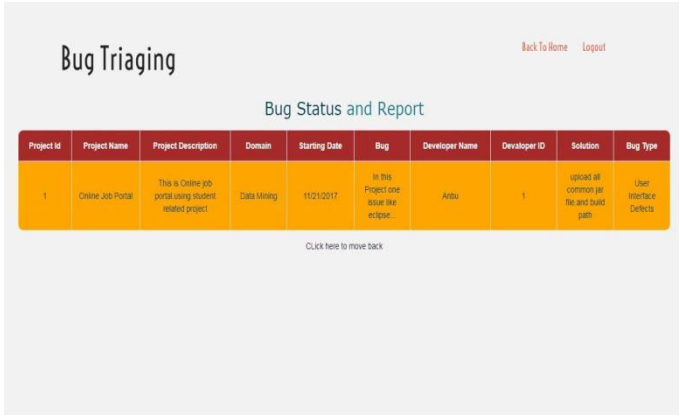
Bug repository

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

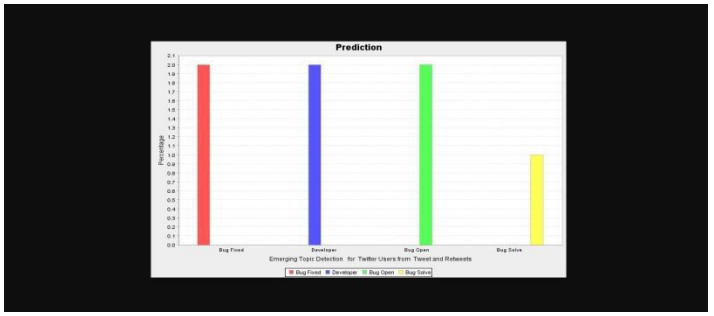
View Bug Request

Admin Home

Bug Status and Report



Over All



VII. CONCLUSION

We propose a new topic model based bug triaging approach, named TopicMiner, and a new topic model, named multi-feature topic model (MTM), which takes into consideration the features of a bug report when assigning topics to words in the report. We have evaluated our solution on 227,278 bug reports from five software systems and demonstrate that TopicMiner MTM outperforms Bugzie, LDA-KL, SVM-LDA, LDA-Activity, and Yang et al.'s approach by substantial margins.

In the future, we plan to improve the effectiveness of our approach further, and

investigate additional bug reports. Also, in this work, we merge the two features (i.e., product and component) as one composite feature (i.e., by creating a feature combination). Other ways of using the multiple features exist and we plan to explore them in a future work. We also plan to design a better topic model to predict fixers when the number of bug reports in a specific product component combination is small (e.g., by using a mixture of models which includes a general model that the approach can back off to when the number of bug reports in a specific product-component combination is small).

REFERENCES

[1] V. Basili, L. Briand, and W. Melo, "A validation of object-oriented design metrics as quality indicators," Software Engineering, IEEE Transactions on, vol. 22, no. 10, pp. 751–761, Oct 1996.

[2] T. Gyim´othy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," IEEE Transactions on Software Engineering (TSE), vol. 31, no. 10, pp. 897–910, 2005.

[3] N. Ohlsson and H. Alberg, "Predicting fault-prone software modules in telephone switchess," Software Engineering, IEEE Transactions on, vol. 22, no. 12, p. 886894, 1996.

[4] N. Nagappan and T. Ball, "Static analysis tools as early indicators of pre-release defect density," in Proceedings of the 27th International Conference on Software Engineering, ser. ICSE '05. New York, NY, USA: ACM, 2005, pp. 580–586. [Online]. Available: http://doi.acm.org/10.1145/1062455.1062558

[5] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in Proceedings of the Third International Workshop on Predictor Models in Software Engineering, ser. PROMISE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 9–. [Online]. Available: http://dx.doi.org/10.1109/PROMISE.2007.10

[6] A. N. Taghi M. Khoshgoftaar, Nishith Goel and J. McMullan, "Detection of software modules with high debug code churn in a very large legacy system," in Software Reliability Engineering. IEEE, 1996, pp. 364–371.

[7] J. S. M. Todd L. Graves, Alan F. Karr and H. P. Siy, "Predicting fault incidence using software change history," Software Engineering, IEEE Transactions on, vol. 26, no. 7, pp. 653–661, 2000.

[8] A. E. Hassan, "Predicting faults using the complexity of code changes," in ICSE. Vancouver, Canada: IEEE Press, 2009, pp. 78–88.

[9] R. Bell, T. Ostrand, and E. Weyuker, "The limited impact of individual developer data on software defect prediction," Empirical Software Engineering, vol. 18, no. 3, pp. 478–505, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10664-011-9178-4

[10] T. J. Ostrand, E. J. Weyuker, and R. M. Bell, "Programmer-based fault prediction," in Proceedings of the 6th International Conference on Predictive Models in

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

Software Engineering, ser. PROMISE '10. New York, NY, USA: ACM, 2010, pp. 19:1–19:10. [Online]. Available: http://doi.acm.org/10.1145/1868328.1868357

[11] R. Moser, W. Pedrycz, and G. Succi, "Analysis of the reliability of a subset of change metrics for defect prediction," in Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ser. ESEM '08. New York, NY, USA: ACM, 2008, pp. 309–311. [Online]. Available: http://doi.acm.org/10.1145/1414004.1414063

[12] R. M. Bell, T. J. Ostrand, and E. J. Weyuker, "Does measuring code change improve fault prediction?" in Proceedings of the 7th International Conference on Predictive Models in Software Engineering, ser. Promise '11. New York, NY, USA: ACM, 2011, pp. 2:1–2:8. [Online]. Available: http://doi.acm.org/10.1145/2020390.2020392

[13] W. P. Raimund Moser and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," in International Conference on Software Engineering (ICSE), ser. ICSE '08, 2008, pp. 181–190.

[14] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," in Proceedings of the 28th International Conference on Software Engineering, ser. ICSE '06. New York, NY, USA: ACM, 2006, pp. 452–461. [Online]. Available: http://doi.acm.org/10.1145/1134285.1134349

[15] M. DAmbros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," Empirical Software Engineering, vol. 17, no. 4, p. 531577, 2012.

[16] J. Sliwerski, T. Zimmermann, and A. Zeller, "Don't program on fridays! how to locate fix-inducing changes," in Proceedings of the 7th Workshop Software Reengineering, May 2005.

[17] L. T. Jon Eyolfso and P. Lam, "Do time of day and developer experience affect commit bugginess?" in Proceedings of the 8th Working Conference on Mining Software Repositories, ser. MSR '11, 2011, pp. 153–162.

[18] F. Rahman and P. Devanbu, "Ownership, experience and defects: a fine-grained study of authorship," in Proceedings of the 33rd International Conference on Software Engineering, ser. ICSE '11, 2011, pp. 491–500.

[19] E. J. W. J. Sunghun Kim and Y. Zhang, "Classifying software changes: Clean or buggy?" IEEE Transactions on Software Engineering (TSE), vol. 34, no. 2, pp. 181–196, 2008.

[20] C. Bird, N. Nagappan, B. Murphy, H. Gall, and P. Devanbu, "Don't touch my code!: Examining the effects of ownership on software quality," in Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ser. ESEC/FSE '11. ACM, 2011, pp. 4–14.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

[21] G. Bavota, B. De Carluccio, A. De Lucia, M. Di Penta, R. Oliveto, and O. Strollo, "When does a refactoring induce bugs? an empirical study," in Proceedings of the 12th International Working Conference on Source Code Analysis and Manipulation, ser. SCAM '12, 2012, pp. 104–113.