International Journal of Computer Applications (0975 – 8887)
Volume 172 – No.6, August 2017

A Survey on Automatic Defect Report Triage

Sayali S. Pawar
ME Student
Department of Computer Engineering, Pune Institute of Computer Technology.
Pune, India.

S. S. Sonawane
Professor
Department of Computer Engineering, Pune Institute of Computer Technology.
Pune, India.

ABSTRACT

Software organizations use defect tracking system to keep track of reported software defects or bugs. Assigning a defect report to the proper developer is called as defect report triage. Large projects receive a large number of defects daily. It is a labor-intensive task to assign these defects to proper developers manually. It is prone to mistakes, like the assignment of defect report to a wrong developer. An automatic approach for defect or bug report triage reduces cost and time required in defect report triage. There are existing machine learning and information retrieval techniques for automatic defect assignment. This paper presents a survey of available defect report triage methods. First this paper gives a brief background of the defect reports then summarizes the existing defect report triage techniques and points out problems with these techniques.

General Terms

Survey, Bug Triage, Software Engineering.

Keywords

Defect/Bug Tracking System, Defect/Bug Report, Defect-Report Triage.

1. INTRODUCTION

Bug or defect tracking plays an important role in software development. Manual defect tracking is difficult and error prone. Defect tracking systems allow developers, testers and customers to submit defects, feature requests and also allow defect fixing. Defect report contains information needed to document, reproduce and fix the defect.

Software industries spend about half of their economy in finding out and fixing defects [2]. Large amounts of defect reports are submitted in open source software projects like Eclipse and Mozilla. Large open source projects like Eclipse receive daily on average 29 defect reports [5]. In practice defects are manually assigned to the developers [11]. Though software defect reports improve the quality of software, manual defect report triage is error prone. It is very tedious to analyze these defect reports and assign them to the proper developers. Often defect reports are of poor quality and are assigned to wrong developers. This may lead to defect report reassignment as currently assigned developer may not deal with the defect report. The process of manual defect report triage is very tedious and time consuming.

The research is going on how to automate the process of defect-report triage. Existing defect report triage methods use machine learning techniques, information retrieval techniques and mathematical techniques such as fuzzy sets and Euclidean distance. Machine learning techniques include Naive Bayes, Support Vector Machine (SVM) and decision trees (C4.5) [4, 8, 9]. In information retrieval techniques defect report is treated as a document having information needed to assign the defect report. These techniques make use of text categorization for triage, and also use feature and instance selection mechanisms to reduce the size of the defect dataset [10]. But due to various reasons, none of the existing techniques are able to achieve the decent accuracy [16]. Due to this, defect reports are still manually triaged. Moreover, existing defect tracking systems are not able to provide any automatic defect-report triage support.

This paper presents the survey on current research on automatic defect report triage methods and discusses problems associated with these methods.

2. DEFECT REPORT

Defect or bug tracking systems have been used since 1970 as a collaboration ecosystem to report and resolve bugs [3]. Defect reports are submitted by users when they encounter a software failure. The software defect is an error, fault or flow in the software which produces unexpected behavior or result. The defect reports generally contain a summary of the defect, the environment settings in which it was triggered and the steps to reproduce it. Once a defect report is created, the development team will try to diagnose and confirm the failure, only then proceeding with its correction [1]. A status is assigned to a defect which allows tracking of the defect. These statuses can be system specific. Once the defect is reported, it will have the status open. The defects, in general, can have three different resolutions: fixed, when the defect has been confirmed and corrected; no fix, when the defect cannot be reproduced or when the defect is not relevant or real defect and system works as per expectations; duplicate, when the defect is duplicate of the another defect which has already been reported; and defer, when the defect is minor and it will be fixed in the future. Unresolved defects remain open. Comments are also very important in defects report as they give additional information about the defect.

Defect reports are also known as bug reports, error reports or fault reports. Typically a defect report consists of the title of the defect, product id, component id, version information, severity of the defect, when it was reported and modified, assigned developer name, resolution status, description of the defect, comments by users and any attachments (test cases, patches). Defect report triage is a process of analyzing a defect report, checking whether it’s a defect or not, checking if there are any duplicates of the defect report, prioritizing defect reports and assigning defect reports to proper developers for fixing.

Open source projects Eclipse and Mozilla use defect tracking systems such as Bugzilla, Jira and Trac. Defect tracking systems allow submission, tracking and management of defects. Defect reports are stored in defect or bug repositories.

3. DEFECT REPORT TRIAGE

3.1 Machine Learning

Several machine learning techniques such as Naive Bayes, Support Vector Machines and decision trees are used in automation of defect report Triage. Instance and feature selection methods are used to reduce the size of the defect repository to discard defect reports of poor quality, unresolved defects and duplicate defects in order to improve the accuracy of defect report triage. These machine learning techniques use text categorization. These techniques train a model on previously assigned defect reports. Then use that model to classify and assign new defect reports.

Cubranic and Murphy [4] proposed the first machine-learning based supervised defect report triage approach. They used history information on developers and fixed defects as training data to assign the developer. They used keywords extracted from title, description and developer's id to train Naive Bayes classifier. They assigned 15859 defect reports and achieved 30% classification accuracy. Later, Anvik et al. improved previous approach by filtering out noisy defect data including defect reports labeled "wontfix" or "worksforme", developers who no longer worked there and fixed less than 9 reports [5, 6]. They compared three classifiers Naive Bayes, SVM and C4.5. SVM performed better than other two approaches and achieved upto 64% accuracy [16]. Their approach achieved higher accuracy than the previous approach (more than 50%).

Xuan et al. [9] first proposed a semi-supervised approach by combining a Naive Bayes classifier and expectation-maximization, which generates a weighted recommendation list for defect-report triage. They used both labeled and unlabeled defect reports. Their showed the classification accuracy can be improved up to 6% than the previous approaches. They concluded the experimental results are not sufficient for real world applications and attributed this to poor defect report quality.

Podgurski et al. [15] also used machine learning techniques to classify defect reports but their approach was more focused on classifying and prioritizing defects than the defect assignment.

To improve the accuracy of defect report triage, Bhattacharya and Neamtiu [7] used rich feature vectors, refined classification and tossing graphs. They observed intranet fold updates are useful for improving defect report triage accuracy.

Hao et al. [22] proposed a defect report triage system called "BugFixer". They used a new tokenization algorithm and Vector Space Model to compute similarity between two defect reports. Their approach outperformed previous techniques based on Naive Bayes and SVM.

All the above approaches have used open-source software projects for defect report triage but Lin et al. [8] carried out defect report triage on Chinese proprietary software. They used SVM on Chinese defect reports. He also carried out another study on the Chinese software using non-text fields of the defect report such as BUGTYPE, BUGCLASS, PHASEID, SUBMITTER, MODULEID, BUGPRIORITY using J48 classifier and decision trees. He observed textual data is more important than non-textual data for automatic defect report assignment. Based on the experimental results, among the three machine learning algorithms Naive Bayes, SVM and C4.5, SVM produces competitive results [16].

All automatic defect report triage techniques focus on assigning correct developers to defect reports but Alenezi and Magel's [12] approach redistributed the load of the overloaded developers.

Feature vectors are important to improve the accuracy of defect report triage. To improve the accuracy defect report triage feature selection techniques are used to reduce the size of the defect dataset. Zou et al. [10] combined feature selection with instance selection to reduce training dataset in order to improve the accuracy of defect report triage. Their approach removed 70% words and 50% defect reports to improve the accuracy.

Park et al. [14] treated defect report assignment problem as an optimization problem to reduce the cost and to increase the accuracy of triage. The proposed cost aware defect report triage algorithm extracts important features from defect reports and trains an SVM model on it. The approach has shown to reduce 30% of the cost.

3.2 Information Retrieval

Information retrieval based techniques treat defect report as a document containing information needed to assign new defect reports. Canfora and Cerulo [21] used descriptions provided in defect reports as a query to find appropriate developers. They created a database of developers. But their approach ignored developers who may have contributed to some code related to the defect but may never have assigned a defect report before. The experienced developers may not always be suitable to handle a defect report.

Term selection techniques in information retrieval are used to improve the accuracy of defect report triage. Alenezi and Magel [12] compared five term-selection techniques. Matter et al. [20] used vocabulary found in developer's source code to create a model expertise of developers. They extracted information from new defect reports and looked it up in the vocabulary to recommend suitable developers. Their approach recommended top-k developers and produced accuracies of 33.6% for top 1 developer and 71% for top 10 developers. Unlike other approaches, it does not depend upon the quality of defect reports and can recommend suitable developers even though they have not worked on any defect reports before.

3.3 Tossing Graphs

Defect tossing is the process of reassigning a defect report to other developer. Jeong et al. [11] proposed a defect tossing graph model to reduce defect tossing. The proposed model has reduced 72% of defect tossing and improved 23% extended their work by using additional attributes on edges and nodes, which has reduced 86.31% of tossing paths and achieve 83.62% of prediction accuracy in defect-report assignment [16].

3.4 Fuzzy Set

Tamrawi et al. [18] used a Fuzzy Set to represent the developers who have the defect-fixing expertise applicable to a specific technical term. The defect reports previously fixed by developers are used to find capable developers to fix the defects. They proposed a tool "Bugzie" that combines fuzzy sets with the tossing graphs. The approach has been evaluated to achieve higher accuracy and efficiency than other work.

3.5 Euclidean Distance

Xia et al. [19] performed two kinds of defect report triage techniques: based on the developers and based on the bug reports. In the developer based technique they measured the distance between a developer and a defect report to recommend suitable developer by considering the defect reports resolved by a developer previously. In the technique based on the defect reports, the previously fixed similar defect reports are used to recommend suitable developers.

4. FEATURES USED IN DEFECT REPORT TRIAGE

Different defect report triage techniques used different features present in a defect report for training a classifier. The selection of features in defect reports plays a crucial role in automatic defect report triage as the accuracy of triage is dependent on the features selected to train a model. Cubranic and Murphy [4] used summary and description of a defect report. Baysal et al. [17] used the summary, the description, and the comments; Bhattacharya et al. [7] and Tamrawi et al. [18] used the bug report ID, the fixing developer ID, and summary as well as the description; Park et al. [14] extracted features from the description of defect reports and its metadata (i.e. version, platform, and target milestone). Anvik et al. [6] removed unresolved, reopened defect reports and defect reports assigned to the developers who no longer worked there or inexperienced developers. Matter et al. [20] used vocabulary found in source code.

5. DISCUSSION

So far machine learning and information retrieval techniques are mainly used to automate the process of defect report triage. Machine learning based techniques use text categorization and train a model on previously assigned defect reports. Information retrieval based techniques use term selection to retrieve information from the defect reports to assign suitable developers. Tossing graph and fuzzy set based techniques are not that common.

Despite the extensive research on defect report triage none of the existing techniques have produced more than 95% of the accuracy [16]. So these techniques are still not used in practice. The defect reports are still manually triaged.

With the recent advancements in deep learning for text categorization, in future deep learning techniques can be used to triage defect reports automatically. Researchers can also focus on improving the accuracy of defect report triage and using new features such as user comments, logs and attachments.

6. CONCLUSION

This paper presented an extensive survey on automatic defect-report triage. Main techniques used for automatic defect-report triage are machine learning and information retrieval based techniques. Machine learning based techniques use text categorization for automatic triage. They first train a classifier on existing defect reports and later the classifier is used to recommend suitable developers for new defect reports. Among three main machine learning techniques Naive Bayes classifier, SVM and decision trees, SVM gives competitive results. Information retrieval based techniques treat defect report as a document containing information needed to predict suitable developers. Feature selection based techniques reduce the size of the training dataset by removing reports of poor quality and reports which are not relevant to improve the accuracy of defect-report triage. Other techniques used in automatic defect-report triage are tossing graphs and Euclidean distance, but these techniques are not that popular. Despite the extensive research in this field, none of the techniques are able to produce the satisfactory accuracy for a real world application and defect reports are still manually triaged.

In future new classification techniques can be explored to improve the accuracy of defect-report triage. Also different features of the defect report can be used to train the classifiers.

7. REFERENCES

[1] J. Aranda and G. Venolia, "The secret life of bugs: Going past the errors and omissions in software repositories", ICSE, 2009.

[2] R. S. Pressman, "Software Engineering: A Practitioner's Approach", 7th ed., New York, NY, USA: McGraw-Hill, 2010.

[3] Rafael Lotufo, Leonardo Passos and Krzysztof Czarnecki, "Towards Improving Bug Tracking Systems with Game Mechanisms", MSR, 2012.

[4] Cubranic D. and Murphy G. C., "Automatic bug triage using text categorization", Proceedings of the International Conference on Software Engineering Knowledge Engineering, Alberta, pp.92-97, 2004.

[5] Anvik J., Hiew L. and Murphy G. C., "Who should fix this bug?", Proceedings of the International Conference on Software Engineering, Shanghai, pp.361-370, 2006.

[6] Anvik J., "Automating bug report assignment", Proceedings of the International Conference on Software Engineering, Shanghai, pp.937-940, 2006.

[7] Bhattacharya P. and Neamtiu I., "Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging", Proceedings of the IEEE International Conference on Software Maintenance, Timisoara, pp.1-10, 2010.

[8] Lin Z., Shu F., Yang Y., et al., "An empirical study on bug assignment automation using Chinese bug data", Proceedings of the International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, pp.451-455, 2009.

[9] Xuan J., Jiang H., Ren Z., et al., "Automatic bug triage using semi-supervised text classification", Proceedings of International Conference on Software Engineering Knowledge Engineering, Redwood City, pp.209-214, 2010.

[10] Zou W., Hu Y., Xuan J., et al., "Towards training set reduction for bug triage", Proceedings of the Annual IEEE International Computer Software and Applications Conference, Munich, pp.576-581, 2011.

[11] Jeong G., Kim S. and Zimmermann T., "Improving bug triage with bug tossing graphs", Proceedings of the joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering, Amsterdam, pp.111-120, , 2009.

[12] Alenezi M., Magel K. and Banitaan S., "Efficient bug triaging using text mining", JSW, Vol. 8(9), pp.2185-2190, 2013.

[13] Xie J., Zhou M. and Mockus A., "Impact of triage: a study of mozilla and gnome", Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, Baltimore, pp.247-250, 2013.

[14] Park J. W., Lee M. W., Kim J., et al., "Costriage: a cost-aware triage algorithm for bug reporting systems", Proceedings of the Conference on Artificial Intelligence, San Francisco, pp.139-144, 2011.

[15] Podgurski A., Leon D., Francis P., Masri W., Minch M., Sun J. and Wang B., "Automated support for classifying software failure reports", ICSE, pp.465-475, 2003.

[16] Zhang Jie, Wang Xiao Yin, Hao Dan, Xie Bing and Zhang Lu MEI Hong, "A survey on bug-report analysis", Sci China InfSci, Vol.58, pp.021101:1-021101:24, February 2015.

[17] Baysal O., Godfrey M. W. and Cohen R., "A bug you like: a framework for automated assignment of bugs", Proceedings of the IEEE International Conference on Program Comprehension, Vancouver, pp.297-298, 2009.

[18] Tamrawi A., Nguyen T. T., Al-Kofahi J., et al., "Fuzzy set-based automatic bug triaging", Proceedings of the International Conference on Software Engineering, Waikiki, pp.884-887, 2011.

[19] Xia X., Lo D., Wang X., et al., "Accurate developer recommendation for bug resolution", Proceedings of the Working Conference on Reverse Engineering, Koblenz, pp.72-81, 2013.

[20] Matter D., Kuhn A. and Nierstrasz O., "Assigning bug reports using a vocabulary-based expertise model of developers", Proceedings of the International Working Conference on Mining Software Repositories, Vancouver, pp.131-140, 2009.

[21] Canfora G. and Cerulo L., "Supporting change request assignment in open-source development", Proceedings of the ACM Symposium on Applied Computing, Dijon, pp.1767-1772, 2006.

[22] Hu H., Zhang H., Xuan J., et al., "Effective bug triage based on historical bug-fix information", Proceedings of the IEEE International Symposium on Software Reliability Engineering, Naples, pp.122-132, 2014.