Report on
‘Not All Bugs Are the Same: Understanding, Characterizing, and Classifying the Root Cause of Bugs’

from

Gemma Catolino, Fabio Palomba, Andy Zaidman and Filomena Ferrucci1

Journal of Systems and Software volume 152

Presented by Pragya Bhandari

Department of Computer Science, Mathematics, Physics and Statistics
Winter Term II
The University of British Columbia
Okanagan

April 12, 2023

1 Introduction

The process of software development is prone to bugs that can cause various problems, such as unexpected crashes, data loss, security vulnerabilities, etc., that can harm the users' experience. Identifying and fixing the root causes of these bugs is crucial, but it can be challenging due to the complexity and diversity of software systems. Unsurprisingly, software maintenance is considered the costliest process in the software development life cycle.

To manage software bugs, development teams use a bug-tracking system, which helps track the bug's status from the moment it's reported to when it's resolved, fixed, and closed. Once a bug ticket is opened, the team decides its priority and assigns it to a developer to be fixed. To ensure efficient bug triaging and assignment, it's helpful to choose the appropriate developer based on their experience working with the relevant features or concepts. However, this process can be time-consuming and requires some knowledge about the bug to be able to select the best resource.

The information in a bug report typically includes a title and a description when first opened that may not always be sufficient to decipher the principal cause of the issue. As a result, manual triaging and assigning bugs may require multiple meetings and readings of the bug report. To address this, Catolino et al. [1] proposed a comprehensive analysis of the fundamental causes of bugs and their distinctive features based on a dataset of bug reports from four open-source software projects. The authors [1] carried out a systematic method for categorizing the root causes of bugs into 9 types. Catolino et al. [1] investigated the distribution and frequency of each bug type, the topics, and the severity of the bugs. Also, Catolino et al. [1] developed and evaluated a machine-learning model that can automatically classify bugs into their root cause categories based on the information in bug reports.

Catolino et al. [1] believed their work would contribute in two ways. First, understanding the root causes of bugs and automatically classifying them would facilitate the efficiency of the debugging process. Second, it would enable teams to assign bugs even more effectively, thus avoiding the occurrence of bug tossing, which happens when defects are allocated to various developers during the bug's lifespan.

In this paper, we present a thorough analysis of the research by Catolino et al. [1] and critique their research choices and strategies. Furthermore, we present our experiments with their dataset, using alternative strategies for topic modeling. We will first explore the related works discussed in the paper [1], then proceed to discuss the main concepts and techniques used in the three research questions proposed by Catolino et al. [1]. We will analyze the results of each experiment and present our critiques on the overall work by Catolino et al. [1].

2 Related Work

Bugs can be classified based on various criteria. However, there are not many studies that have explored bug classes based on the root causes of a bug. In this section, we will review a few related works that have addressed the bug classification problem.

IBM proposed the first and most well-known taxonomy for bug classification, called Orthogonal Defect Classification (ODC) [2]. This classification system includes 13 categories that enable software developers to classify defects based on their impact on the user, focusing on their consequences rather than their underlying cause. Another common bug categorization method was established by Hewlett-Packard [4], which employs three characteristics to depict defects: “origin,” which indicates the activity in which the issue originated; “mode” which explains the conditions leading to a bug; and “type” which offers additional information on the position of a bug by specifying if it is linked to hardware or software.

Among the related works, the research work most similar to that of Catolino et al.[1] is that of Tan et al. [7]. Tan et al. proposed a study that investigated the distribution and impact of three fundamental causes of bugs (semantic, security, and concurrency issues) in software systems like Apache, Mozilla, and Linux. They also proposed a machine learning approach to automatically classify these types of bugs with an average F-Measure of about 70%. However, the article [1] we are scrutinizing endeavors to offer a more comprehensive perspective on the underlying origins of defects and their propagation across a larger array of systems (119 compared to 3) without any initial speculation. The objective of the study [1] is to construct a defect categorization system that can be broadly applied, and to establish a root cause classification model that can automatically classify all of the root causes that have been identified.

3 Methodology

In this section, we will discuss the dataset, methods, and strategies used by the paper [1] respective to each of the research questions they aimed to address.

3.1 Dataset

The dataset used by Catolino et al.[1] comprises 1280 issue reports from three software ecosystems: Mozilla, Apache, and Eclipse. These systems were chosen based on previous studies [8] that reported their bug reports as complete and understandable, thus making them the ideal choice for analysis. The authors of the paper [1] randomly sampled 1,280 bug reports that were fixed and closed and excluded misclassified bug reports that did not contain actual bugs. The dataset includes defect reports from 119 projects within the ecosystems. The dataset was first manually analyzed and classified to build a taxonomy of classification as part of the first research question which we have discussed below.

3.2 RQ1: To what degree can the information provided in bug reports be used to group and categorize the root causes of bugs?

3.2.1 Approach

The paper [1] utilized an iterative process to develop a taxonomy of bug root causes. The process involved analyzing 1,280 bug reports from the dataset, and categorizing them based on root cause. The taxonomy was developed through three iterations of content analysis, each involving an initial batch of bug reports, followed by a discourse to refine the taxonomy. The final taxonomy comprised of 9 categories, which were validated by involving 5 industrial developers with more than 10 years of programming experience. Each of the individuals involved in the study found the classification system to be lucid and comprehensive, and the labels they designated were identical to the ones selected during the taxonomy creation stage. The authors then used this taxonomy to perform more analysis on understanding patterns within the categories which are discussed in the second research question.

3.3 RQ2: What are the recurring patterns and features of different bug categories, such as how often they occur and the topics words associated with them?

This research question uses two types of analysis. The authors[1] conducted a frequency analysis on the dataset to investigate the distribution of bug classification and their occurrence rates in software projects. Additionally, the authors utilized topic analysis on the bug summaries to gain a better understanding of the main topics and themes addressed in the bug reports.

3.3.1 Approach

Frequency Analysis: After producing a labeled dataset of bug reports with their classification type, Catolino et al. [1] performed frequency analysis to analyze the distribution of the classification types in the dataset. The paper [1] did not specify any special techniques used for frequency analysis so it can be assumed that the analysis was a plain data analysis of the classified dataset. Nevertheless, it is worth mentioning that in the research outlined in [1], a defect could not be allocated to more than one category because of the refinement level of their categorization system.

Topic Analysis: To understand the characteristics of each of the bug root cause category, the authors of the study [1] aimed to ascertain the prevalent subjects addressed in the bug reports corresponding to each category. They [1] employed Latent Dirichlet Allocation (LDA), a renowned method for topic modeling. LDA is a technique for clustering based on topics, which is used to cluster documents in the topic vector space by comparing their topic distributions. The authors adopted the following steps for each bug category in their taxonomy:

- They extracted all the terms that made up the bug reports in a certain category.
- An Information Retrieval (IR) normalization process was applied to the extracted terms. The process included spelling correction, contraction expansion, nouns and verbs filtering, and singularization. The terms were then transformed by applying camel case splitting, lower case letters reduction, special characters, programming keywords, and common English stop words removal, and stemming using Porter's stemmer.
- The processed terms were provided as input to the LDA-GA algorithm created by Panichella et al. [5]. This method is an improved variation of the regular LDA technique that employs a genetic algorithm to strike a balance between the coherence of topics and the distinctiveness of clusters in order to determine the optimal number of clusters to produce from the input terms.

After understanding the characteristics relating to the classifications, Catolino et al. [1] proceeded to implement an automatic bug classification model based on the root causes as their third research question.

3.4 RQ3: How well does their bug classification model perform when using bug report data to identify the root causes of bugs?

3.4.1 Approach

The authors [1] focused on assessing the feasibility of a classification model for classifying bug root causes using bug reports to address the research question. Catolino et al. [1] used machine learning algorithms to perform experiments with. Catolino et al.[1] used summary messages from bug reports as independent variables in their root cause classification model. However, the textual contents in a bug summary might not be representative enough to isolate the root cause of a bug. To solve this issue, the researchers [1] employed the Term Frequency - Inverse Document Frequency (TF-IDF), a scoring technique that determines the relative frequency of words in a particular textual document compared to the inverse proportion of that word over the entire document corpus. In addition to TF-IDF, the researchers also experimented with two other feature extraction methods i.e., Doc2Vec and Word2Vec. After the feature extraction step, the researchers [1] experimented with different classifiers such as Naive Bayes, Support Vector Machines (SVM), Logistic Regression, and Random Forest to provide a wider overview of the performance achieved by different classifiers. For the hyperparameter optimization step, the Grid Search algorithm was used, a brute force strategy to estimate machine learning approach hyperparameters by testing the Cartesian product of all the potential hyperparameter values for each classifier. The paper [1] reported that Logistic Regression provided the best performance and thus have only reported detailed results for Logistic Regression model alone. To validate the model, Catolino et al. [1] adopted 10-fold cross-validation, applying stratified sampling to ensure each fold had the same proportion of each bug class. The Synthetic Minority Oversampling Technique (SMOTE) was used to make the training data balanced in case of training data imbalance.

4 Results

In this section the results of each of the research questions have been presented.

4.1 RQ1: Taxonomy of Bug root causes

After the manual labeling of bug reports, the dataset resulted in bug reports falling into 9 different categories based on their root causes. The different categories and their brief description are listed below:

- Configuration Issue: These issues consisted of bugs caused due to problems in external files or the incorrect paths to directories or files.

- Network Issue: These issues mainly include the bugs caused due to connection or server errors relating to network problems, communication protocols, or unanticipated server shutdowns.
- Database-related Issue: These issues contained all bugs relating to connection problems between the database and the software application.
- GUI-related Issue: These issues included all front-end related bugs, for example, styling issues or errors with the front-end elements' behaviors that fall strictly in the front-end part of the application.
- Performance Issue: This bug category includes bugs having performance issues, including memory overuse, energy leaks, and functions causing non-terminated loops.
- Permission/Deprecation Issue: Errors categorized in this group can be attributed to two primary factors. Firstly, they stem from deprecated method calls or APIs being present, modified, or removed. Secondly, complications arising from unused API permissions are also encompassed within this category.
- Security Issue: This category comprises bugs related to vulnerabilities and other security-related issues. These bugs are generally associated with reloading certain parameters and removing unused permissions that may decrease the system's overall reliability.
- Program Anomaly Issue: Bugs arising from implementing new features and are specifically concerned with exceptional circumstances, such as problems with return values and unexpected crashes due to logic issues rather than the graphical user interface (GUI), are included in this category. It should be noted that bugs resulting from incorrect SQL statements do not belong to this category. Instead, the bugs have been classified as database-related issues defined earlier.
- Test Code-related Issue: These issues span problems that occur as a result of (i) running, modifying, or updating test cases, (ii) tests that operate inconsistently, and (iii) test limitations that prevent them from discovering bugs that occur in various parts of the software.

4.2 RQ2: Characteristics of Bug types

4.2.1 Frequency Analysis

The frequency distribution of each bug classification type was observed and has been summarized in Table 1. The most frequent type of bug was found to be Program Anomaly issue types which are caused by logical code failures, which is expected as most bugs are caused while trying to implement some new functionality. The second most frequent bug type is GUI-related issue which is consistent with most software engineering expectations as new design changes or front-end implementations would prominently cause these types of bugs. The third most frequent type of bug was found to be Configuration issue which includes bugs caused by failures in external libraries and APIs which are widely used in systems as tools to accelerate the development process.

Issue type	**Frequency**
Program Anomaly issue	41%
GUI-related issue	17%
Configuration issue	16%
Test Code-related issue	7%
Performance issue	4%
Permission/Deprecation issue	4%
Security issue	4%
Network issue	4%
Database-related issue	3%

Table 1: Results of Frequency Analysis [1]

Categories	**Topic 1**	**Topic 2**	**Topic 3**	**Topic 4**	**Topic 5**
Program Anomaly issue	error	file	crash	exception	-
GUI-related issue	page	view	render	select	font
Configuration issue	link	file	build	plugin	jdk
Test Code-related issue	fail	test	retry	-	-
Performance issue	thread	infinite	loop	memory	-
Permission/Deprecation issue	deprecated	plugin	goal	-	-
Security issue	security	xml	packageaccess	vulnerable	-
Network issue	server	connection	slow	exchange	-
Database-related issue	database	sql	connection	connection	-

Table 2: Topic Analysis result generated using LDA-GA [1]

4.2.2 Topic Analysis

The topic analysis produced representative topic words for each category with few overlaps. The results obtained from the topic analysis process have been organized in Table 2. For each category, Catolino et al. [1] tried to produce 5 topic words, however for some categories LDA-GA algorithm could not produce enough topic words and hence the results contain some missing topics.

Each category has closely associated topic words that accurately describe that category. For instance, the GUI-related issue has words like "*page*" and "*render*". However, some topic words collection are not qualitatively optimal for example in Database-related issue category there are repeating topic word "*connection*".

4.3 RQ3: Automatic Bug Classification Model

The authors of the study [1] found that among the experiments performed with different text embeddings and machine learning models, the best overall performance was recorded for TF-IDF embeddings and Logistic Regression classifier model. The evaluation metrics chosen for the classification model were Precision, Recall, F1-Score and AUC-ROC. The results from the combination of TF-IDF and Logistic Regression classifier model as reported in the paper

Class	**Precision**	**Recall**	**F1-score**	**AR**
Configuration issue	46	52	49	68
Database-related issue	71	63	67	72
GUI-related issue	61	68	65	77
Network issue	36	40	38	56
Performance issue	67	57	62	65
Permission/Deprecation issue	86	55	67	69
Program Anomaly Issue	68	65	67	74
Security issue	76	74	75	88
Test Code-related issue	90	70	79	93
Overall	**67**	**60**	**64**	**74**

Table 3: Table from Classification Report Results [1]

[1] have been tabulated in Table 3.

5 Discussion

In this section, we will discuss the methodological decisions and the results obtained in the paper. We will present our critiques on the paper in two parts, the topic modeling process and the replication package-related concerns.

The authors of the paper [1] chose LDA-GA as their topic modeling algorithm. The authors justify choosing this particular variation of LDA but fail to justify why they selected LDA to begin with. This motivated us to compare their results from the paper [1] with the topic results from the latest topic modeling techniques. We chose BERTopic for this comparison as a paper by Egger and Yu [3] reported that BERTopic gave better results than LDA in topic modeling problems. The results from experimenting with BERTopic are presented in Table 4. The colored cells in the table represent the common topic words between BERTopic and the paper's [1] topic results using LDA-GA. One of the first observations that sets BERTopic apart from LDA-GA is the larger number of topic words found for each category, which LDA-GA lacked for some categories. Comparing the quality of the topic words is a subjective matter; however, some lists of topic words generated by BERTopic seem to be of better quality; for example, there are distinct topic words for the Database-related issue category instead of repeating terms like those shown in Table 2.

Furthermore, the incomplete replication package was a major drawback we faced in analyzing the paper [1] and verifying their results. The replication package neither contained the final dataset generated after their taxonomy building process, nor the updated code that performed the experiments the results of which they have reported in the paper [1]. This threatens their credibility and the validity of their work, and also causes their replication package to be practically unusable by future researchers who would want to extend on their [1] work.

Categories	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Program Anomaly issue	patch	file	crash	exception	agent
GUI-related issue	html	link	render	table	panel
Configuration issue	link	server	build	plugin	review
Test Code-related issue	failure	test	junit	regression	ignore
Performance issue	thread	infinite	loop	memory	leak
Permission/Deprecation issue	deprecated	removal	hibernation	transfer	adapt
Security issue	security	cocoon	package	server	file
Network issue	server	build	redirect	abort	connection
Database-related issue	database	queries	select	search	patch

Table 4: Topic Analysis result generated using LDA-GA

6 Future Work

Bugs can be classified into different types based on various criteria. In the paper by Catolino et al. [1], the classification criterion was root cause. Recent studies have shown that bugs can also be classified into intrinsic and extrinsic types based on their causation's origin. Intrinsic and extrinsic bugs were defined in the paper by Rodriguez-Perez et al. [6]. Intrinsic bugs can be traced to a bug causation point in the version control system. In contrast, extrinsic bugs are caused by changes external to the project's scope and control and hence cannot be traced back to a bug-inducing change in the version control system.

We propose that it would be enlightening to observe the distribution of intrinsic and extrinsic bugs for each of the 9 categories based on root causes. For example, we conjecture that bugs from categories like Program Anomaly issues and GUI-related issue types should have more intrinsic bugs, whereas the bugs from categories like Configuration issues and Permission/Deprecation issue types should have more extrinsic bugs. The hypothesis stems from the understanding that intrinsic bugs would be caused by bugs caused by logical errors during new feature implementation. In contrast, extrinsic bugs would be caused due to failures in external files or libraries used in the project's code. We believe it would be an interesting extension of the bug classification problem presented in the paper [1] and potentially might bring forth another pattern that might aid in the automatic bug classification model.

7 Conclusion

Bug classification is essential to bug analysis, that might help in bug triaging and then ultimately in bug mitigation processes. The paper [1] proposed that *Not All Bugs are the Same* and hence, should not be treated the same either. The paper [1] aimed to develop a taxonomy of bug root causes and a root cause prediction model for bug reports in software projects belonging to three large ecosystems: Mozilla, Apache, and Eclipse. The study analyzed 1,280 bug reports and identified 9 different root causes behind the bugs. The root causes were studied from 2 angles: frequency of appearance and principal topics in corresponding bug reports. The implemented root cause classification model performed well in terms of F-Measure, and AUC-ROC with average scores of 64% and 74% respectively when tested using a 100 times 10-fold cross-validation methodology. The study [1] concluded that the proposed taxonomy and root cause prediction model could help improve the bug reporting and fixing process in software development.

References

[1] Catolino, G., Palomba, F., Zaidman, A., and Ferrucci, F. Not all bugs are the same: Understanding, characterizing, and classifying the root cause of bugs, 2019.

[2] Chillarege, R., Bhandari, I., Chaar, J., Halliday, M., Moebus, D., Ray, B., and Wong, M.-Y. Orthogonal defect classification-a concept for in-process measurements. *IEEE Transactions on Software Engineering 18*, 11 (1992), 943–956.

[3] Egger, R., and Yu, J. A topic modeling comparison between lda, nmf, top2vec, and bertopic to demystify twitter posts. In *Front Sociol. 2022 May 6* (2022).

[4] Freimut, B., Denger, C., and Ketterer, M. An industrial case study of implementing and validating defect classification for process improvement and quality management. In *11th IEEE International Software Metrics Symposium (METRICS'05)* (2005), pp. 10 pp.–19.

[5] Panichella, A., Dit, B., Oliveto, R., Di Penta, M., Poshynanyk, D., and De Lucia, A. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *2013 35th International Conference on Software Engineering (ICSE)* (2013), pp. 522–531.

[6] Rodríguez-Pérez, G., Nagappan, M., and Robles, G. Watch out for extrinsic bugs! a case study of their impact in just-in-time bug prediction models on the openstack project. *IEEE Transactions on Software Engineering 48*, 4 (2022), 1400–1416.

[7] Tan, L., Liu, C., Li, Z., Wang, X., Zhou, Y., and Zhai, C. Bug characteristics in open source software. *Empirical Softw. Engg. 19*, 6 (dec 2014), 1665–1705.

[8] Zimmermann, T., Premraj, R., Bettenburg, N., Just, S., Schröter, A., and Weiss, C. What makes a good bug report? *IEEE Transactions on Software Engineering 36* (09 2010), 618–643.