ATLANTIS PRESS

Advances in Engineering Research, volume 123

2nd International Conference on Materials Science, Machinery and Energy Engineering (MSMEE 2017)

Identifying misclassified bug reports

Suo Hu[1], Zhou Zou[2, *]

[1]School of Computer and Science, Hubei Engineering University, Xiaogan, Hubei 432000, PR China

[2]Discipline Construction Office, Hubei Engineering University, Xiaogan, Hubei 432000, PR China

[*]Corresponding author: zouzhou@hbeu.edu.cn

Keywords: bug reports, classification rules, misclassification.

Abstract The data mining methodology for identification and detection of bugs is an important application. Especially separating bugs from non-bugs is a general challenge. When software developers classify bug reports, they may misclassify bug reports with bias and errors. All issue reports are analyzed by combining the classification rules from open-ihm project and Herzig et al. (2012). A comparison on the classification results of different authors has been extracted which shows the misclassification rate. Depending on the percent of misclassification rates, it is concluded that the classification of bugs in Herzig et al. (2012) can be applied to the open-ihm project and it exhibited the similar proportion of misclassified bugs as reported in Herzig et al. (2012).

Introduction

When we talk about bug in Information Technology area, it is not a kind of insect, but it represents an error or fault in the software which displays invalid output or wrong results. In general, most bugs are produced from errors in software and human mistakes. Almost each software or program will contain a large number of bugs, therefore, when users find bugs, they will report bugs to developers. Nowadays it is becoming normal to analyze bug databases to figure out where bugs have occurred in the past time and predict how it will occur in the future. In bug databases, there are some types of bugs. For instance, they are arithmetic bugs, interfacing bugs, team working bugs, syntax bugs, logic bugs and so on. Bugs reports always request for semantically changes to source code. Additionally, other issue reports request different solutions such as fixing code, adding new features, updating documentation, internal refactoring and so on. In this way, if the request is not about fixing errors in the code so that it will be considered as a non-bug issue.

Open-ihm is an open source software project which contributes to data and information collection from poverty families. It aims to help developers and researchers to analyze and record the income data of household. The purpose of building open-ihm is to use information technology to everywhere of the world. Open-ihm is knows as an innovative and reliable data collection method and it can be an open and quick way to collect the data. The significant issue is that open-ihm software is required to assist experts and non-experts in the analysis of data using specified models.

The total classification process in Herzig et al (2012) will be carried out in four steps. In the first stage, the first author will analyze all the issue reports and assign to category by using the classification rules. In the second stage, my supervisor will classify these issue reports again without knowing the results of the first stage. Next, both authors will compare their classified issue reports and detect the classification conflicts-that is, find out the difference in the classification results. Finally, re-classify the different results by a joint pair-inspection of both authors with the comments of issue reports and classification rules.

Here are two research questions which will be the aims of this project:

Can the classification of bugs in Herzig et al. (2012) be applied to the open-ihm project?

Does the open-ihm project exhibit the similar proportion or misclassified bugs as reported in Herzig et al. (2012)?

Summary of Literature Review

How do bugs get reported. According to Nicolas Bettenburg et al. (2008) indicates the way to make a good bug report. On one hand bug reports are vital evidence to indicate problems of software, on the other hand bug reports contains the details of description about the errors as well as the location. However, there are different qualities of bug reports on the websites. Bettenburg et al mentions that the quality of bug reports will be investigated in the view of developers. The aim of it is to find out the factors which impact the quality of bug reports. A questionnaire has been applied and the conclusion is "there is a mismatch between what developers consider most helpful and what users provide". It means for both users and developers they all have different views on which factors impact the bug reports most. The requirements of bug reports should provide some important and sensitive information to suit both reporters and developers.

According to Simon Tatham (1999), the purpose of this article is to figure out the way to report bugs effectively. Actually the object of bug reports is to help the developers to realize which parts of the software go wrong. Tatham indicates that "many bug reports provide nothing or give the wrong information". In order to improve this issue, he suggests users either show the program failures in person, or provide the detailed stages on how you made it failed. Some comments like "it does not work" or "I cannot tell how it goes wrong" will not be useful. The journal shows what users should care and which part needs to be avoiding when reporting bugs. He suggests users describe every step in details and give their own comments and solutions as possible.

How do bugs get classified. There is a set of classification rules and categories in Herzig et al. (2012). There are six categories of classification and they are BUG, RFE, IMPR, DOC, REFAC and OTHER. The definitions of these are from Herzig et al. (2012).

BUG: means Fix Request, definition is "Issue reports documenting corrective maintenance tasks that require semantically changes to source code."

RFE: means Feature Request, definition is "Issue reports documenting an adaptive maintenance task whose resolving patch(es) implemented new functionality."

IMPR: means Improvement Request, definition is "Issue reports documenting a perfective maintenance task whose resolution improved the overall handling or performance of existing functionality."

DOC: means Documentation Request, definition is "Issue reports solved by updating external (e.g. website) or code documentation (e.g. JavaDoc)."

REFAC: means Refactoring Request, definition is "Issue reports resolved by refactoring source code. Typically, these reports were filed by developers."

OTHER: means Other Request, definition is "Any issue report that did not fit into any of the other categories. This includes: reports requesting a back port (BACKPORT), code cleanups (CLEANUP), changes to specification (rather than documentation or code; SPEC) general development task (TASK), and issues regarding test cases (TEST)."

How do bugs get misclassified. According to Herzig et al. (2012), an issue report will be classified as bug if it requests for corrective code maintenance. However, some issue reports request for "perfective and adaptive maintenance, refactoring, discussions, requests for help, and so on" (Antonio et al. 2008) which do not request for code maintenance, and would not be classified as bugs. In addition, Antonio et al (2008) mentioned that some issue reports are classified as bugs, but essentially they are referring to non-bug issues. Issue reports are classified by developers manually so incorrect empirical developers will produce bias in parts of confusing reports. Therefore, the result of the classification reports is not accurate.

Summary of Research Methodology

In this project we have combined the two different classification rules from open-ihm and Herzig et al. (2012). We will introduce the stages of classification bug reports and show how to analyse the data set.

Data set. In this project the aim of research is to examine if the classification bug reports in Herzig et al. (2012) can be used in open-ihm. And observe if it can result in similar proportion of misclassified bugs as reported in Herzig et al. (2012). So the data set are bug reports in open-ihm project of Google code which are all reported by users and developers.

Classification rules in Herzig et al. (2012). According to Herzig et al. (2012), it demonstrates the classification categories and rules of issue reports. As these issue reports are collected from five projects and chosen from RESOLVED, CLOSED, VERIFIED and FIXED.

Classification in open-ihm Issue reports in open-ihm project are allocated unique ID and many labels. When a new issue is being reported, first users need to consider this issue is a defect or a new feature request, then users should fill in the summary of the issue and description about some details. The most important label is Type which will be classified in this research project. Therefore, there is a table to introduce the category of Type in open-ihm.

Short name	Long name	Explanation	Classification rules
Defect	Type-Defect	Report of a software defect	Same as BUG
Enhancement	Type-Enhancement	Request for enhancement	Same as RFE and IMPR
Task	Type-Task	Work item that does not change the code or documentations	Same as OTHER (Sub-category of OTHER)
Review	Type-Review	Request for a source code review	Same as REVIEW
Other	Type-Other	Some other kind of issue	Same as OTHER

Table 3.2 The classification categories and rules of Google Code

Merged classification rules in this project. The merged classification rules will be applied into this project so that we need ensure they are in same categories. In this way, we need to transform the category. The new rules and previous categories in different projects are shown in table 3.3.

New category	Category in Herzig et al. (2012)	Category in open-ihm	New classification rules
BUG	BUG	Defect	Same as BUG in Herzig et al. (2012)
RFE	RFE	Enhancement	Same as RFE in Herzig et al. (2012)
IMPR	IMPR	Enhancement	Same as IMPR in Herzig et al. (2012)
DOC	DOC	Other	Same as DOC in Herzig et al. (2012)
REFAC	REFAC	Other	Same as REFAC in Herzig et al. (2012)
REV	OTHER	Review	Same as Review in open-ihm
OTHER	OTHER	Task, Other	Combine OTHER in Herzig et al. (2012) and Task in open-ihm

Table 3.3 The new classification rules in this project

Classification phases. According to Herzig et al. (2012) we have conducted the manual classification rules in four phases:

In the first phase, one person acts as the first author who will inspect all the issue reports and classify them into different categories using the new classification rules.

In the second phase, another person acts as the second author who will do the same work again without knowing the classification result done by first author.

In the third phase, two copies of classification results from the first author and the second author will be compared to detect classification conflicts. This means, find out the different results from phase one and phase two.

In the last phase, discuss the classification conflicts—issue reports and make them resolved by a joint pair-inspection of both authors. In the re-classification phase, all the comments of issue reports and classification rules should be considered in depth.

The first and second phase of the inspection will be processed individually and both authors know nothing about the classification results from each other. This ensures all the issue reports will be classified equally.

Flow chart. A flow chart has been designed to show all the steps in this project below.

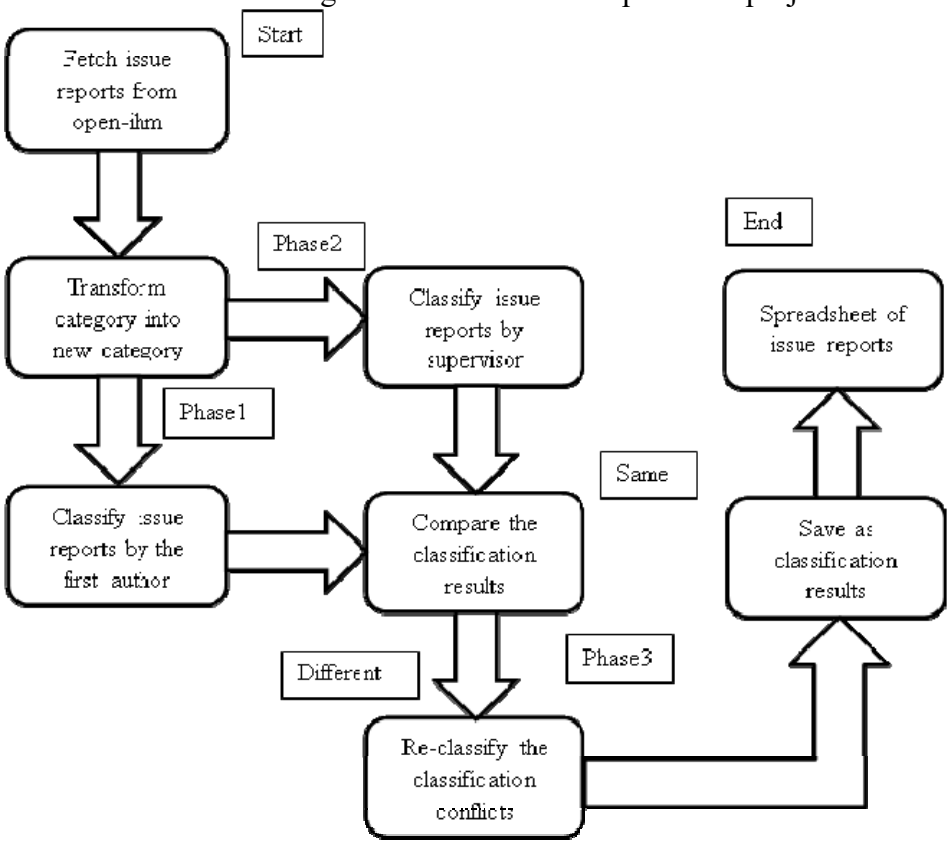

Fig. 3.4 Flow chart in this project

Discussion of Findings

The primary aim of this chapter is to demonstrate the final classification results and analyze the results. So we abandoned the classification results from both authors. In Herzig et al. (2012) among 7401 issue reports there are 3093 reports need to be reclassified. In this project among 273 issue reports there are 142 reports will be re-classified. In this project we have analyzed all 273 issue reports from open-ihm and calculated them into different categories before the classification.

Original category	Amount of category
Defect	177
Enhancement	73
Task	15
Review	2
Other	6
Total	273

Table 4.1 Category in open-ihm

And the original category should be transformed into new category in order to better classification.

New category	Amount of category
BUG	177
RFE	69
IMPR	2
DOC	3
REFAC	3
REV	2
OTHER	17
Total	273

Table 4.2 New category classification

Below are the charts showing final classification results of each category. (Fig. 4.3-4.7)

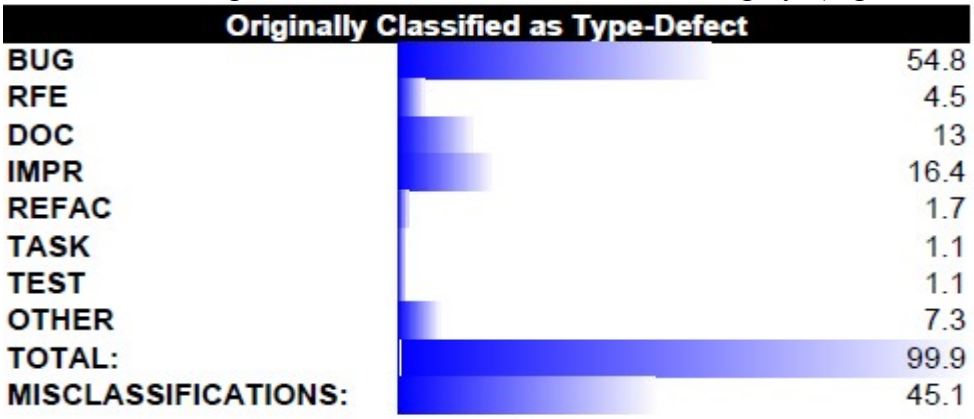

Fig. 4.3 Final classification results of original Defect

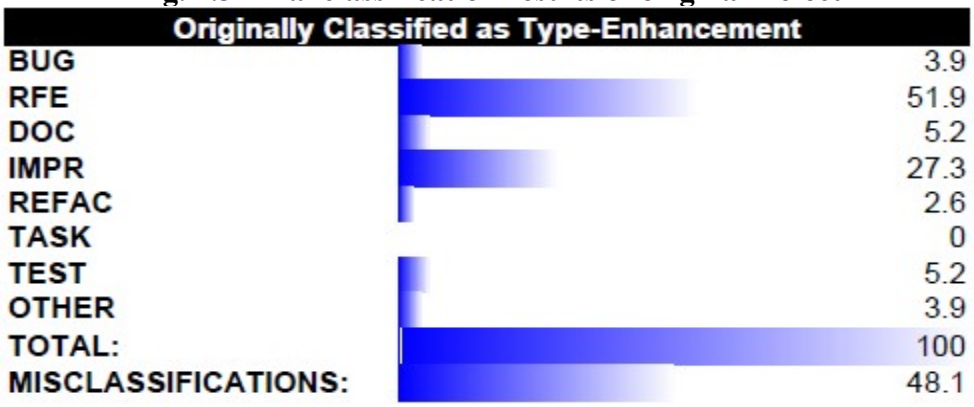

Fig. 4.4 Final classification results of original Enhancement

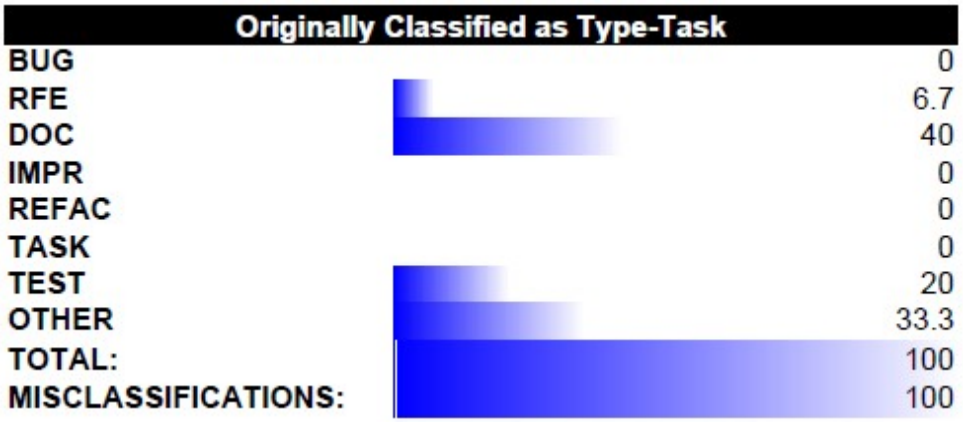

Fig. 4.5 Final classification results of original Task

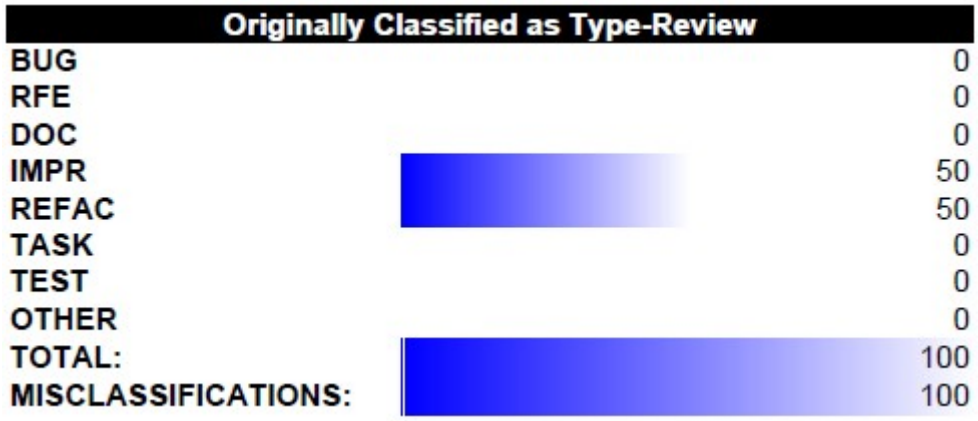

Fig. 4.6 Final classification results of original Review

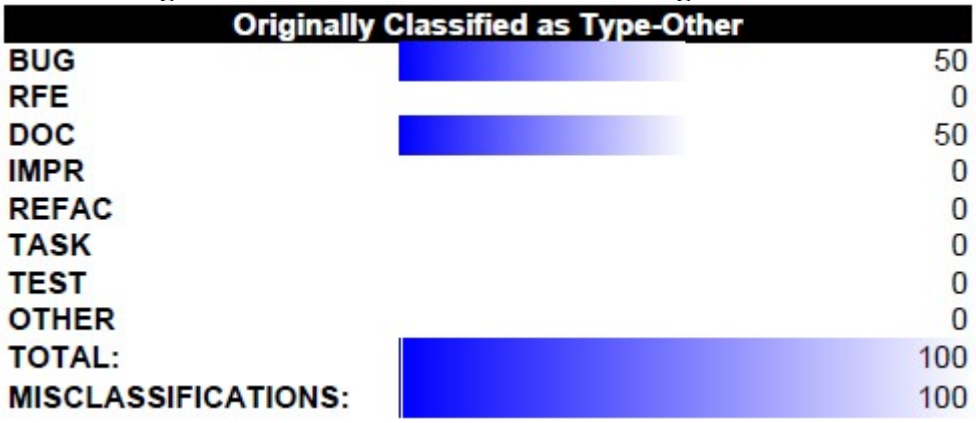

Fig. 4.7 Final classification results of original Other

From the classification results and the conclusions in Herzig et al. (2012) some conclusions have been made:

Issue report classification are unreliable: In this project all 273 issue reports (including open issues) have been classified and 135 of them are found to be misclassified---that is, the total misclassification rate is nearly 50%.

Almost every second bug is not a bug: 45.1% of all bug reports do not refer to corrective code maintenance.

Feature Request and Improvement Request are easily mixed up: In Type-Enhancement there are 51.9% of reports are Feature Request and 27.3% of them are Improvement Request.

Task, Review and Other are easily misclassified: The misclassification rates of Task, Review and Other are 100%.

Summary

The classification of bug reports is a way to help developers find out where the errors locate in software. And deal with bug reports requires human effort and experience. However, we still get some limitations to prevent it running better. First we could not rule out all the errors in manual inspection. That means, developers would get bias as the restriction of their experience and knowledge. Then we need to improve the classification rules more perfect because we found the definitions are not very clear between Feature Request and Improvement Request. The last one is the way we merge the classification conflicts.

As Herzig et al. (2012) indicated “our motivation for this work was to have a well-classified set of bug reports and features, which we now can leverage (and share) for future research”. We have some views on future predicting bug reports and automatic classification. However, it still requires more bug databases and more experience to achieve. Additionally, we suggest we could use the same format of categories when we reporting bugs so that it will be easier to share the bug reports among all the databases.

The misclassification will impact some related studies which use the data sets directly without validation. The misclassification rates can reflect the data quality of bug databases. As we know, the reason of misclassification is that there are different views on bug classification between users and developers. In many cases users report issue reports may not know the difference between improvement, feature request or bug. So the main work for developers is to assign these issue reports. When an issue report gets misclassified, it means we have not figured out the request in this issue report so that the solution probably will not be effective and useful. As Herzig said, "misclassification also impacts the relative order of the most defect-prone files", in this way, it will be more difficult for bug prediction.

References

[1] E. S. Raymond. (1991). *The New Hacker's Dictionary*, The MIT Press, Cambridge.

[2] GIGER, E., PINZGER, M. and GALL, H., (2010) *Predicting the fix time of bugs*, 2010, ACM, pp. 52-56.

[3] Grottke, M., & Trivedi, K. S. (2005). *A classification of software faults. Journal of Reliability Engineering Association of Japan*, 27(7), 425-438.

[4] HANGAL, S. and LAM, M.S., (2002) *Tracking down software bugs using automatic anomaly detection*, 2002, IEEE, pp. 291-301.

[5] HOVEMEYER, D. and PUGH, W., (2004) *Finding bugs is easy*. ACM SIGPLAN NOTICES, 39(12), pp. 92-106.

[6] Howden, W. E. (2005) *Software test selection patterns and elusive bugs*. In Computer Software and Applications Conference, 2005. COMPSAC 2005. 29th *Annual International*, IEEE, 1, pp. 25-32.

[7] J. Gray. (1985). *Why do computers stop and what can be done about it*? Technical Report 85.7, PN87614.

[8] Herzig, K., Just, S., & Zeller, A. (2012). *It's not a Bug, it's a Feature: How Misclassification Impacts Bug Prediction.*

[9] Wikipedia (2013). Software bug.

[10] Tatham, S. (1999). *How to report bugs effectively*. Ultimo acesso, 13.

[11] ZIMMERMANN, T., PREMRAJ, R., BETTENBURG, N., JUST, S., SCHROTER, A. and WEISS, C., (2010) *What Makes a Good Bug Report*? Institute of Electrical and Electronics Engineers, Inc, pp. 618-643.

[12] Weiß, C., Premraj, R., Zimmermann, T., & Zeller, A. (2006). *Predicting effort to fix software bugs. Issues*, 11.

[13] Emerson, M., Thomas, Z., Christian, B. and Nachiappan, N., (2013) *The Design of Bug Fixes*.

[14] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Gu´eh´eneuc, (2008) Is it a bug or an enhancement? A text-based approach to classify change requests, *in Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, 23, pp. 304–318.