2017 IEEE International Conference on Software Quality, Reliability and Security (Companion Volume)

Empirical Study on the Correlation between Software Structural Modifications and Its Fault-proneness

Fei Wang, Jun Ai, Jiaming Wang
School of Reliability and Systems Engineering
Beihang University
Beijing, China
{feiwang, aijun, wjm1992}@buaa.edu.cn

***Abstract*—The rapidly increasing application of software contributes to the growing requirement of software quality. However, current software systems are far from being defect-free. Bugs are the root causes for various software/system failures. Although many rigorous quality assurance techniques are applied in the entire software life cycle, new bugs could be unintendedly injected during the software development process. Therefore, improving software quality by providing many practical indicators for fault prediction is significant. In this study, we performed an empirical study on 873 versions of 4 open-source projects to evaluate the degree of correlation between software structural changes and bugs. Our results could serve as a guide for software testing and development and a powerful tool for software fault prediction.**

***Keywords*—software bugs, software complex networks, structural modifications, fault prediction**

I. Introduction

At present, software plays an important role in some of humanity's complex systems. However, recent high-profile accidents due to software failures remind practitioners that software systems are far from being bug-free. To reduce the risks incurred by bugs effectively and prevent system failures, software components with high degree of fault-proneness should be identified in advance to ensure that necessary actions (e.g., performing an in-depth testing on such software modules) are timely performed. Thus, researchers and practitioners have paid special attention in the areas of bug distribution and fault-proneness prediction [2, 3, 4, 5, 6, 7]. However, several obstacles have been revealed to limit the availability of these techniques in producing considerable practical analysis results.

First, existing publications on fault-proneness prediction focus on modules/files. Although these publications could provide engineers with informative evidence regarding the potential locations of bugs, the granularity of these studies is relatively high and could not provide in-depth information to serve as accurate indicators for fault-proneness prediction. Second, software structural modifications are rarely considered in existing literature. If the relationship between a software component and other components exhibits significant changes, then bugs could be highly introduced by these changes because of the high volatility of software structures. Therefore, we proposed the research question: What is the relationship between structural modifications and the fault-proneness of a function?

II. Overview

In this section, we briefly introduce the procedures utilized in our study, which mainly include the construction of software networks, the determination of bug locations, the identification of structural modifications, and data analysis. The methods of the construction of software networks and the determination of bug locations in paper [1] are carried out in our study. Here we mainly introduce the identification of structural modifications.

With respect to software structural changes, we considered the following two categories of structural modifications:

- In_Degree change (*Deg-In*). With respect to a specific function, *Deg-In* represents that functions calling this function has been changed when compared with the prior version.
- Out_Degree change (*Deg-Out*). Similar to *Deg-In*, the *Deg-Out* of a specific function represents that the functions called by this function have been changed when compared with the prior version.

III. Subject Programs

In this study, we analyzed four open-source projects selected from GitHub, which are text editing software. TABLE I. presents the information of the analyzed projects.

TABLE I. Basic Information For Subject Programs

Software	Vers	Commits	Total Funcs	Total Calls
Nagioscore	98	2665	3510	13337
Macvim	108	15690	6274	19877
Redis	190	6137	6447	22846
Nginx	477	6040	3381	10983

I. Results and Discussion

With respect to a specific category of structural modifications, say *X*, we divided the functions of each version into four distinct groups, represented as $Func_{XB}, Func_{X\bar{B}}, Func_{\bar{X}B}$, and $Func_{\bar{X}\bar{B}}$. $Func_{XB}$ includes the functions that have a structural modification in *X* and contain at least one bug; $Func_{X\bar{B}}$ represents functions with a fluctuation in *X* values but do not possess any bug; $Func_{\bar{X}B}$ contains buggy functions that have no change in their X values; and $Func_{\bar{X}\bar{B}}$ contains only correct functions with no changes in X values.

With the four groups generated, $N_{X,B}, N_{X,\bar{B}}, N_{\bar{X},B}$, and $N_{\bar{X},\bar{B}}$ represent the number of functions within each group, where N denotes the total number of functions within a specific version. Similarly, we defined

$$\begin{cases} N_X = N_{X,B} + N_{X,\bar{B}} \\ N_{\bar{X}} = N_{\bar{X},B} + N_{\bar{X},\bar{B}} \\ N_B = N_{X,B} + N_{\bar{X},B} \\ N_{\bar{B}} = N_{X,\bar{B}} + N_{\bar{X},\bar{B}} \end{cases}.$$

P_X is the proportion of functions with certain structural modifications in the total functions, which can be calculated as follows:

$$P_X = \frac{N_X}{N}. \tag{2}$$

Moreover, of all the faulty functions, $P_{X,B|B}$ demonstrates the proportion of faulty functions that exhibit the structural modifications of type X, as expressed as follows:

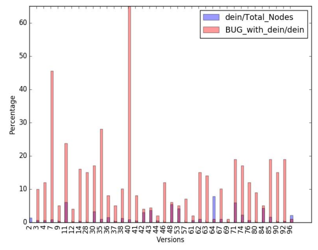
$$P_{X,B|B} = \frac{N_{X,B}}{N_B}. \tag{3}$$

In this section, we excluded the versions that are bug-free or modification-free. We only included the results for Nagioscore, as shown in Fig. 1, due to space limitation. However, this condition does not imply that the results are only applicable to Nagioscore. Similar phenomena can also be observed from the other three software systems (i.e., Redis, Macvim, and Nginx).

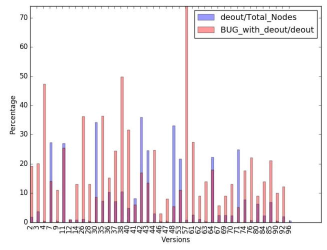
The same evaluation criteria utilized in [1] was followed; thus, if $P_{XB|B}$ is significantly greater than P_X, then bugs are expected to reside in functions with structural modifications of type X. Here, the *Deg-In* of functions is considered as an example. Without loss of generality, assume that for a specific version of program, $P_{\mathrm{Deg\_In},B|B}$ is 0.90, whereas P_{Deg_In} is 0.20. This condition indicates that although only 20% of all the functions in this version are *Deg-In*, 90% of the bugs are actually attributed to the *Deg-In* modification of functions.

In Fig. 1, the blue and red bars represent P_X and $P_{XB|B}$, respectively, whereas the purple bars represent the overlapping of two colors. As shown in Fig. 1, the red bars are significantly higher than blue bars, which indicates that $P_{XB|B}$ is significantly greater than P_X for most of the cases.

Subsequently, SVM algorithm was used to further validate the correlation between the bugs and structural modifications. The *Deg-In* and *Deg-Out* of the functions were utilized as the learning vectors to predict the bug attribute of the functions. The classification accuracy of Nagioscore, Macvim, Redis, and Nginx is 82.25%, 78,94%, 74.43%, and 71.65%, respectively. Thus, the distribution of bugs and structural modifications are significantly correlated.



(a) *Deg-In*



(b) *Deg-Out*

Fig. 1. Correlations between fault-proneness and certain structural modification of functions

I. CONCLUSION

Based on our analysis, we get that bugs are closely related to the structural modifications of the functions. The results could serve as a guide for software testing and development and provide new metric parameters for software fault prediction.

However, some limitations are still found in this paper. The research focused on structured software. Therefore, for future studies, more types of software should be studied

REFERENCES

1. Zhang S, Ai J, Li X. Correlation between the Distribution of Software Bugs and Network Motifs[C]// IEEE International Conference on Software Quality, Reliability and Security. IEEE, 2016:202-213.
2. J. Juran, Quality Control Handbook, McGraw-Hill, New York, 1974.
3. N. Ohlsson, H. Alberg, Predicting fault-prone software modules in telephone switches, IEEE Trans. Softw. Eng. 22 (12) (1996) 886 – 894.
4. B. Compton, C. Withrow, Prediction and control of ADA software defects, J.Syst. Softw. 12 (3) (1990) 199 – 207
5. G. Denaro, M. Pezzè, An empirical evaluation of fault-proneness models, in:Proceedings of the 24th International Conference on Software Engineering (ICSE ’ 02), pp. 241 – 251.
6. M. English, C. Exton, I. Rigon, B. Cleary, Fault detection and prediction in an open-source software project, in: Proceedings of the 5th International Conference on Predictor Models in Software Engineering
7. T. Galinac Grbac, P. Runeson, D. Huljenic, A second replicated quantitative analysis of fault distributions in complex software systems, IEEE Trans. Softw. Eng. 39 (4) (2013) 462–476. .