An industrial study on the differences between pre-release and post-release bugs

Renaud Rwemalika, *University of Luxembourg*, renaud.rwemalika@uni.lu

Issues caused by software defects are a common source of business failures and economic losses. According to the Software Fail Watch of Tricentis in 2017, software bugs resulted in $1.7 trillion of industrial revenue losses and, more importantly, the number of bugs reported increased by 10 percent compared to 2016. These figures highlight the importance of a good understanding on the nature of software bugs and their root causes. On another hand, source code analysis techniques are typically developed and evaluated using some bug instances, which should reflect specific characteristics and assumptions around the targeted bugs. Thus, bug finding and removal techniques, such as software testing, static analysis, fault localization or program repair, are developed according to the characteristics, nature and fixes of bug datasets such as SIR, Defects4j and Bugs.jar. This is a good first step towards developing feasible and effective techniques. However, our perception, understanding and assessment of these techniques is strongly connected to the characteristics of the bugs involved in these datasets.

To understand the bug characteristics and their consequences on software testing and debugging techniques, we perform an extensive study on pre- and post-release bug characteristics of our industrial partner, BGL BNP Paribas. We focus on 'critical' systems, developed in Java. These systems have been audited and tested using both unit and system (end-to-end) test practices. Our aim is to understand the nature of common kinds of real software issues by performing a fine-grained analysis on the recorded bugs and their patches.

In contrast to the majority of previous research that is based on Open Source projects, our study focuses on real industrial systems where code development often differs in architecture, process and people involved. We extract and study both quantitatively, using source code metrics, and qualitatively, observing code context and the properties of the pre- and post-release bug patches. Our industrial partner has established quality assurance teams and procedures, making the distinction between pre- and post-release bugs meaningful. As such, our analysis can help interpreting the feasibility of existing methods/studies, judging the representativeness of dataset used in previous work, help positioning and choosing appropriate pre- or post-release bug data and increases the general understanding of the industrial software issues.

Perhaps the most interesting result from our study is that we find evidence that most of the bug patches we analyzed, especially the post-release ones, involve approximately 82% of additions. This finding suggests that the related bugs fall in the category of the so-called 'omission' bugs. This is particularly important for software testing researchers since omission bugs are a limitation of the widely used and researched code-based software testing techniques (e.g., code coverage. As code-based testing is driven by the existing code, targeting the coverage of codebases, it is hard to reveal issues related to code that is not there.

Another finding regards the scope of the changes required to fix post-release bugs. While fixes requiring complex changes that spread across multiple files exist, the majority of the bugs are fixed locally (with changes applied to the same unit) and in conditional statements. This is good news for testing research and specifically unit testing, as it provides evidence that unit testing could be adequate for targeting such post-release bugs (bugs causes spread across different units being harder to triggered by unit testing).

One last interesting finding we can mention regards the existence of configuration bugs and the relative differences between pre- and post-release bugs. We find that 45.97% and 66.69% of the pre- and post-release bugs require changes on configuration files. We find that post-release bugs require chunks of changes twice as big as pre-release ones and these changes mainly involve control flow modifications (while the pre-release ones involve interface changes). Regarding the locations of fixes, we found that both pre- and post-release locations are similar indicating that the difference are mainly on the nature of bugs than in their location.

Our primary contribution is to raise the awareness of the research community for the need to distinguish and control between pre- and post-release bugs, and to present the results of an industrial empirical study demonstrating significant differences. The most important finding from this control study is the evidence that almost all post-release bugs are 'local' and the apparent existence of the so-called 'omission' bugs. These findings suggesting that future research should focus on unit-based (local) analysis techniques targeting this particular class of bugs, which is, as we discussed, fundamentally different from the rest bug types. Overall, more research is needed in this important area to fully understand this fundamental aspect of software bugs, and we certainly do not claim to have completely answered all questions in this paper. We do, however, believe that our findings significantly improve the understanding of bugs nature, the structural differences between pre- and post-release bugs, and the areas where source code analysis techniques should focus on.

Work initially presented at 35th IEEE International Conference on Software Maintenance and Evolution DOI: 10.1109/ICSME.2019.00019