Watch out for Extrinsic Bugs! A Case Study of their Impact in Just-In-Time Bug Prediction Models on the OpenStack project

*Presentation abstract

Gema Rodríguez-Perez
University of Waterloo
Waterloo, Canada
gemrodper@gmail.com

Mei Nagappan
University of Waterloo
Waterloo, Canada
mei.nagappan@uwaterloo.ca

Gregorio Robles
Universidad Rey Juan Carlos
Madrid, Spain
grex@gsyc.urjc.es

I. ABSTRACT

In a previous paper [1], we have shown how bugs can be of different nature: they can be intrinsic or extrinsic bugs. **Intrinsic bugs** are bugs that were introduced by one or more specific changes to the source code. They are bugs for which a "bug-introducing change" (BIC) can be identified in the version control system of a software system. In contrast, **extrinsic bugs** are bugs that were introduced by changes not registered in the version control system (e.g., from an external dependency), or changes in requirement. Therefore, extrinsic bugs do not have an explicit BIC in the version control system.

Generally, researchers in **Just-In-Time *(JIT)* bug prediction** have considered all bugs to be intrinsic bugs. That means, for each bug analyzed, researchers assumed to be able to identify a specific change in the control version system that introduced the bug. This paper [2] shows an example of how considering the existence of extrinsic bugs can affect software engineering research. Specifically, we study the impact of extrinsic bugs in JIT bug prediction by partially replicating a recent study by McIntosh and Kamei on JIT bug prediction [3].

The main contributions of this paper are: i) over 40% of bugs in a dataset that was extracted automatically are not intrinsic bugs; ii) item JIT models obtain different performance in terms of Area Under the Curve (AUC) (up to 16 percent AUC points) when they consider only intrinsic bugs; iii) AUC scores are more stable after removing extrinsic bugs; iv) mislabeled bugs affect JIT models reducing their performance up to 4 percent AUC points; and v) Intrinsic and extrinsic bugs have different code change properties. When analyzing mislabeled bugs, we found that the *nature* of extrinsic bugs is closer to mislabeled than to intrinsic bugs.

We first analyzed whether a manually curated dataset without extrinsic bugs differs from and automatic extracted dataset **(RQ1)**. We then studied the impact of extrinsic bugs in JIT models by adding a constraint (i.e., "when extrinsic bugs are removed") to McIntosh and Kamei's original research questions [3] **(RQ2-RQ4)**. As we found a significant share of mislabeled bugs in McIntosh and Kamei's dataset, we also analyzed the impact of mislabeled bugs **(RQ5)**. Mislabeled bugs refer to issue reports that have been considered as bug reports when, in fact, they are not reporting a bug but another software maintenance activities, e.g., enhancements or refactoring. Finally, we studied whether intrinsic, extrinsic, and mislabeled bugs have different characteristics **(RQ6)**.

Our results indicate the negative role that extrinsic bugs have on the performance of JIT approaches. When removing extrinsic bugs from the trained data used in OpenStack, JIT models obtain a more accurate representation of the real world as indicated by their different (often higher) AUC values in their performance. These models capture change properties better. Therefore, JIT models that are fitted only with intrinsic bugs obtain more stable AUC scores and lose less predictive power.

We also offer evidence that extrinsic bugs are of different nature than intrinsic bugs. Actually, they are more similar to issues that are not bugs than to intrinsic bugs. We think that this finding is not only relevant for JIT bug prediction models, but that it may impact many other areas of software engineering practice and research, and would like to call for further research on extrinsic bugs. An important implication of our paper is that researchers and practitioners should be aware of the data that feed JIT bug prediction models. Although with the current state of the art data validation might be tedious and very labor-intensive to achieve, researchers should be aware that considering extrinsic bugs might impact results.

REFERENCES

[1] G. Rodríguez-Pérez, G., Robles, G., Serebrenik, A., Zaidman, A., German, D. M., and Gonzalez-Barahona, J. M., "How bugs are born: a model to identify how bugs are introduced in software components", Empirical Software Engineering, vol. 25, no 2, p. 1294-1340, Mar 2020.

[2] G. Rodríguez-Pérez, M. Nagappan, and G. Robles, "Watch Out for Extrinsic Bugs! A Case Study of Their Impact in Just-In-Time Bug Prediction Models on the OpenStack Project," IEEE Trans. Softw. Eng., vol. XX, pp. x–x, September 2020.

[3] S. McIntosh and Y. Kamei, "Are fix-inducing changes a moving target? a longitudinal case study of just-in-time defect prediction," IEEE Trans. Softw. Eng., vol. 44, no. 5, pp. 412–428, May 2018.