On the Diffuseness and the Impact on Maintainability of Code Smells: A Large Scale Empirical Investigation

Fabio Palomba[1], Gabriele Bavota[2], Massimiliano Di Penta[3]
Fausto Fasano[4], Rocco Oliveto[4], Andrea De Lucia[5]
[1]University of Zurich, Switzerland – [2]Universitá della Svizzera Italiana, Switzerland – [3]University of Sannio, Italy
[4]University of Molise, Italy – [5]University of Salerno, Italy
palomba@ifi.uzh.ch,gabriele.bavota@usi.ch,dipenta@unisannio.it
fausto.fasano@unimol.it,rocco.oliveto@unimol.it,adelucia@unisa.it

ABSTRACT

Code smells were defined as symptoms of poor design choices applied by programmers during the development of a software project [2]. They might hinder the comprehensibility and maintainability of software systems [5]. Similarly to some previous work [3, 4, 6, 7] in this paper we investigate the relationship between the presence of code smells and the software change- and fault-proneness. Specifically, while previous work shows a significant correlation between smells and code change/fault-proneness, the empirical evidence provided so far is still limited because of:

Limited size of previous studies: The study by Khomh *et al.* [4] was conducted on four open source systems, while the study by D'Ambros *et al.* [1] was performed on seven systems. Furthermore, the studies by Li and Shatnawi [6], Olbrich *et al.* [7], and Gatrell and Counsell [3] were conducted considering the change history of only one software project.

Detected smells *vs.* manually validated smells: Previous work studying the impact of code smells on change- and fault-proneness relied on data obtained from automatic smell detectors, whose imprecisions might have affected the results.

Lack of analysis of the magnitude: Previous work indicated that some smells can be more harmful than others, but the analysis did not take into account the magnitude of the observed phenomenon. For example, even if a specific smell type may be considered harmful when analyzing its impact on maintainability, this may not be relevant in case the number of occurrences of such a smell type in software projects is limited.

Lack of analysis of the magnitude of the effect: Previous work indicated that classes affected by code smells have more chances to exhibit defects (or to undergo changes) than other classes. However, no study has observed the magnitude of such changes and defects, *i.e.*, no study addressed the question: *How many defects would exhibit on average a class affected by a code smell as compared to another class affected by a different kind of smell, or not affected by any smell at all?*

Lack of within-artifact analysis: A class might be intrinsically change- and/or fault-prone, *e.g.*, because it plays a core role in the system. Hence, the class may be intrinsically "smelly". Instead, there may be classes that become smelly during their lifetime because of maintenance activities. Or else, classes where the smell was removed, possibly because of refactoring activities. For such classes, it is of paramount importance to analyze the change- and fault-proneness of the class during its evolution, in order to better relate the cause (presence of smell) with the possible effect (change- or fault-proneness).

Lack of a temporal relation analysis: While previous work correlated the presence of code smells with high fault- and change-proneness, one may wonder whether the artifact was smelly when the fault was introduced, or whether the fault was introduced before the class became smelly.

To cope with the aforementioned issues, this paper aims at corroborating previous empirical research on the impact of code smells by analyzing their diffuseness and effect on change- and fault-proneness on a total of 395 releases of 30 open source systems, considering 13 different code smell types manually identified. Our results showed that classes affected by code smells tend to be significantly more change- and fault-prone than classes not affected by design problems, however their removal might be not always beneficial for improving source code maintainability.

ACM Reference Format:
Fabio Palomba[1], Gabriele Bavota[2], Massimiliano Di Penta[3] Fausto Fasano[4], Rocco Oliveto[4], Andrea De Lucia[5]. 2018. On the Diffuseness and the Impact on Maintainability of Code Smells: A Large Scale Empirical Investigation. In *ICSE '18: ICSE '18: 40th International Conference on Software Engineering , May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3180155.3182532

ICSE '18, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5638-1/18/05.
https://doi.org/10.1145/3180155.3182532

REFERENCES

[1] M. D'Ambros, A. Bacchelli, and M. Lanza. On the impact of design flaws on software defects. In *Quality Software (QSIC), 2010 10th International Conference on*, pages 23–31, July 2010.
[2] M. Fowler. *Refactoring: improving the design of existing code.* Addison-Wesley, 1999.
[3] M. Gatrell and S. Counsell. The effect of refactoring on change and fault-proneness in commercial c# software. *Science of Computer Programming*, 102(0):44 – 56, 2015.
[4] F. Khomh, M. D. Penta, Y.-G. Guéhéneuc, and G. Antoniol. An exploratory study of the impact of antipatterns on class change- and fault-proneness. *Empirical Software Engineering*, 17(3):243–275, 2012.
[5] P. Kruchten, R. L. Nord, and I. Ozkaya. Technical debt: From metaphor to theory and practice. *IEEE Software*, 29(6):18–21, 2012.
[6] W. Li and R. Shatnawi. An empirical study of the bad smells and class error probability in the post-release object-oriented system evolution. *Journal of Systems and Software*, pages 1120–1128, 2007.
[7] S. M. Olbrich, D. S. Cruzes, and D. I. K. SjÃÿberg. Are all code smells harmful? a study of god classes and brain classes in the evolution of three open source systems. In *in: IntâĂŹl Conf. Softw. Maint*, pages 1–10, 2010.