The Scent of a Smell: An Extensive Comparison between Textual and Structural Smells

Fabio Palomba[1], Annibale Panichella[2], Andy Zaidman[2], Rocco Oliveto[3], Andrea De Lucia[4]
[1]University of Zurich, Switzerland – [2]Delft University of Technology, The Netherlands
[3]University of Molise, Italy – [4]University of Salerno, Italy
palomba@ifi.uzh.ch,a.panichella@tudelft.nl,a.e.zaidman@tudelft.nl
rocco.oliveto@unimol.it,adelucia@unisa.it

ABSTRACT

Code smells, *i.e.*, symptoms of poor design and implementation choices applied by programmers during the development of a software project [2], represent an important factor contributing to technical debt [3]. The research community spent a lot of effort studying the extent to which code smells tend to remain in a software project for long periods of time [9], as well as their negative impact on non-functional properties of source code [4, 7]. As a consequence, several tools and techniques have been proposed to help developers in detecting code smells and to suggest refactoring opportunities (*e.g.*, [5, 6, 8]).

So far, almost all detectors identify code smells using structural properties of source code. However, recent studies have indicated that code smells detected by existing tools are generally ignored (and thus not refactored) by the developers [1]. A possible reason is that developers do not perceive the code smells identified by the tool as actual design problems or, if they do, they are not able to practically work on such code smells. In other words, there is misalignment between what is considered smelly by the tool and what is actually refactorable by developers.

In a previous paper [6], we introduced a tool named TACO that uses textual analysis to detect code smells. The results indicated that textual and structural techniques are complementary: while some code smell instances in a software system can be correctly identified by both TACO and the alternative structural approaches, other instances can be only detected by one of the two [6].

In this paper, we investigate whether code smells detected using textual information are as difficult to identify and refactor as structural smells or if they follow a different pattern during software evolution. We firstly performed a repository mining study considering 301 releases and 183,514 commits from 20 open source projects (i) to verify whether textually and structurally detected code smells are treated differently, and (ii) to analyze their likelihood of being resolved with regards to different types of code changes, *e.g.*, refactoring operations. Since our quantitative study cannot explain relation and causation between code smell types and maintenance activities, we perform a qualitative study with 19 industrial developers and 5 software quality experts in order to understand (i) how code smells identified using different sources of information are perceived, and (ii) whether textually or structurally detected code smells are easier to refactor. In both studies, we focused on five code smell types, *i.e.*, *Blob*, *Feature Envy*, *Long Method*, *Misplaced Class*, and *Promiscuous Package*.

The results of our studies indicate that textually detected code smells are perceived as harmful as the structural ones, even though they do not exceed any typical software metrics' value (*e.g.*, lines of code in a method). Moreover, design problems in source code affected by textual-based code smells are easier to identify and refactor. As a consequence, developers' activities tend to decrease the intensity of textual code smells, positively impacting their likelihood of being resolved. Vice versa, structural code smells typically increase in intensity over time, indicating that maintenance operations are not aimed at removing or limiting them. Indeed, while developers perceive source code affected by structural-based code smells as harmful, they face more problems in correctly identifying the actual design problems affecting these code components and/or the right refactoring operation to apply to remove them.

ACM ISBN 978-1-4503-5638-1/18/05.
https://doi.org/10.1145/3180155.3182530

ACM Reference Format:
Fabio Palomba[1], Annibale Panichella[2], Andy Zaidman[2], Rocco Oliveto[3], Andrea De Lucia[4]. 2018. The Scent of a Smell: An Extensive Comparison between Textual and Structural Smells. 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/10.1145/3180155.3182530

REFERENCES

[1] G. Bavota, A. De Lucia, M. Di Penta, R. Oliveto, and F. Palomba. An experimental investigation on the innate relationship between quality and refactoring. *Journal of Systems and Software*, 107(9):1–14, Sept. 2015.
[2] M. Fowler. *Refactoring: improving the design of existing code.* Addison-Wesley, 1999.
[3] P. Kruchten, R. L. Nord, and I. Ozkaya. Technical debt: From metaphor to theory and practice. *IEEE Software*, 29(6):18–21, 2012.
[4] F. Palomba, G. Bavota, M. Di Penta, F. Fasano, R. Oliveto, and A. De Lucia. On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. *Empirical Software Engineering*, pages 1–34, 2017.
[5] F. Palomba, G. Bavota, M. Di Penta, R. Oliveto, D. Poshyvanyk, and A. De Lucia. Mining version histories for detecting code smells. *IEEE Transactions on Software Engineering*, 41(5):462–489, May 2015.
[6] F. Palomba, A. Panichella, A. De Lucia, R. Oliveto, and A. Zaidman. A textual-based technique for smell detection. In *International Conference on Program Comprehension*, pages 1–10, May 2016.
[7] D. I. K. Sjøberg, A. F. Yamashita, B. C. D. Anda, A. Mockus, and T. Dybå. Quantifying the effect of code smells on maintenance effort. *IEEE Transactions on Software Engineering*, 39(8):1144–1156, 2013.
[8] N. Tsantalis and A. Chatzigeorgiou. Identification of move method refactoring opportunities. *IEEE Transactions on Software Engineering*, 35(3):347–367, 2009.
[9] M. Tufano, F. Palomba, G. Bavota, R. Oliveto, M. Di Penta, A. De Lucia, and D. Poshyvanyk. When and why your code starts to smell bad (and whether the smells go away). *IEEE Transactions on Software Engineering*, 43(11):1063–1088, 2017.