The scent of a smell: an extensive comparison between textual and structural smells

Palomba, Fabio; Panichella, Annibale; Zaidman, Andy; Oliveto, Rocco; De Lucia, Andrea

DOI
10.1145/3180155

Publication date
2018

Document Version
Accepted author manuscript

Published in
Proceedings of the 40th International Conference on Software Engineering (ICSE)

Citation (APA)

Important note
To cite this publication, please use the final published version (if applicable). Please check the document version above.
The Scent of a Smell: An Extensive Comparison between Textual and Structural Smells

Fabio Palomba¹, Annibale Panichella², Andy Zaidman², Rocco Oliveto³, Andrea De Lucia⁴
¹University of Zurich, Switzerland – ²Delft University of Technology, The Netherlands
³University of Molise, Italy – ⁴University of Salerno, Italy
palomba@ifi.uzh.ch,a.panichella@tudelft.nl,a.e.zaidman@tudelft.nl
rocco.oliveto@uninol.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 structural 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 Reference Format:

REFERENCES