Information and Software Technology 000 (2017) 1–13

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

How developers perceive smells in source code: A replicated study

Davide Taibi*, Andrea Janes, Valentina Lenarduzzi

Free University of Bozen-Bolzano, Piazza Università 1, 39100 Bolzano, Italy

ARTICLE INFO

Article history:
Received 12 December 2016
Revised 16 August 2017
Accepted 17 August 2017
Available online xxx

Keywords:
Software maintenance
Code smells
Bad smells
Antipatterns
Refactoring

ABSTRACT

Context. In recent years, smells, also referred to as bad smells, have gained popularity among developers. However, it is still not clear how harmful they are perceived from the developers' point of view. Many developers talk about them, but only few know what they really are, and even fewer really take care of them in their source code.

Objective. The goal of this work is to understand the perceived criticality of code smells both in theory, when reading their description, and in practice.

Method. We executed an empirical study as a differentiated external replication of two previous studies. The studies were conducted as surveys involving only highly experienced developers (63 in the first study and 41 in the second one). First the perceived criticality was analyzed by proposing the description of the smells, then different pieces of code infected by the smells were proposed, and finally their ability to identify the smells in the analyzed code was tested.

Results. According to our knowledge, this is the largest study so far investigating the perception of code smells with professional software developers. The results show that developers are very concerned about code smells in theory, nearly always considering them as harmful or very harmful (17 out of 23 smells). However, when they were asked to analyze an infected piece of code, only few infected classes were considered harmful and even fewer were considered harmful because of the smell.

Conclusions. The results confirm our initial hypotheses that code smells are perceived as more critical in theory but not as critical in practice.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Software Quality Assurance (SQA), i.e., assuring that software fulfills the posed quality standards, remains a task that requires effort and expertise.

Software is invisible, which "means that it is very easy for the project to proceed for a considerable time before problems become apparent, and without it being possible to verify that the passing of time and expenditure of money correlate with progression of the project in the desired direction [1]." To cope with the invisibility of software is costly: quality-related information is difficult to collect [2,3], requires time and effort; as a consequence, often other activities, e.g., adding new features to a product are given a higher priority then investing in improving the internal quality of the software [3]. Quality tools aim to reduce the costs of quality, however, while such tools alleviate the task of collecting quality-related information, they often require substantial effort to understand the analysis they provide.

One type of analysis that quality tools provide, is the detection of code smells [4] and antipatterns [5]. Code smells are structural characteristics of software, which may indicate code or design problems that can make software hard to evolve and maintain [4]. In this work we adopt the term code smells for both code smells and antipatterns.

Several studies consider code smells harmful from a maintenance point of view [6–11], while others suggest that smells are not terribly problematic [12]. Code Smells are also considered a cause of potential faults by several studies [13–17], while other studies report a significant but small effect on them [18]. Moreover, Code Smells present in the source code are also considered causes of higher change-proneness [14,15,19–21] and low code understandability [22].

Developers often do not know about the code smells they introduce into their source code (including the decreasing maintainability). For this reason, the identification of code smells (and the investment of time to remove them) is gaining acceptance in industry [23].

* Corresponding author.
E-mail addresses: davide.taibi@unibz.it (D. Taibi), andrea.janes@unibz.it (A. Janes), valentina.lenarduzzi@unibz.it (V. Lenarduzzi).

http://dx.doi.org/10.1016/j.infsof.2017.08.008
0950-5849/© 2017 Elsevier B.V. All rights reserved.

Some examples of SQA tools that are proposing techniques for detecting and reducing code smells are JDeodorant,[1] DECOR [24], or SonarQube.[2] However, the term "code smell" is not always clearly understood. As an example, SonarQube, one of the most frequently used tools for SQA, renamed the concept of "Code Issues", which originally related to the adherence to coding standards, into the term "code smells", increasing the misunderstanding of code smells from the developers' point of view. In this context, in addition to the question of whether SQA practices are worth the effort or not, researchers are also discussing how developers perceive code smells:

- if they are perceiving them as a serious problem that deserves extra effort to be solved; and
- if they see them as some sort of hint that they will take into consideration next time they will edit that code.

The first question that arises is whether developers have a common understanding what a code smell is; the second how harmful they consider them.

Several researchers have investigated code smells in the past; we focus on two particular studies that studied the two questions just mentioned: the first study by Yamashita and Moonen [8] entitled "Do Developers Care about Code Smells? An Exploratory Survey" investigates if developers consider code smells as something harmful; the second study by Palomba et al. [25] entitled "Do they Really Smell Bad? A Study on Developers' Perception of Bad Code Smells" investigates if developers have a common understanding of code smells, i.e., if when confronted with them in the code they recognize it as the same problem, and how harmful they see the problem. At the best of our knowledge, these two studies were the only ones assessing the perceived harmfulness of code smells.

We designed this study as a differentiated external replication of two studies [26]. The differences of this replication to the previous two studies are:

- The respondents in the study by Yamashita and Moonen [8] were mostly originating from India, USA, Pakistan and Romania (38 from 73 respondents). Since we conducted our study during a conference that took place in Europe (see Section 3), we mainly had European and American participants. Moreover, the origin of the respondends in [25] was not reported so we cannot compare this aspect to our study.
- Palomba et al. [25]. interviewed 34 participants in their study. Only 9 were industrial developers, 10 were developers involved in the original projects from which the authors took code examples, and 15 were master students. The selected interviewees expose this study to two threats to validity:
  - Developers who are familiar with the code being evaluated might have a different perception of the harmfulness[3] of a code smell than somebody who has never seen the code. Some code smells may be intentionally left in the code because they are a side effect of another design decision. Such background knowledge can bias the elicitation of the attitude of developers towards code smells.
  - The question of whether students can be used as subjects in software engineering experiments is widely debated (e.g., in [27]). We find that in the study on how code smells are perceived, experience plays a crucial role. If the question is how developers perceive code smells, students might assess problematic Java classes differently from professional software developers.
- Palomba et al. consider only 12 code smells, while we extended our study to all 23 code smells proposed by Fowler et al. [4].

According to our knowledge, this is the largest study so far investigating the perception of code smells with professional software developers: we collected the perception about the harmfulness of code smells of 63 participants and assessed the ability to identify and categorize code smells in source code examples of 41 participants. Moreover, we compared the perceived harmfulness of 32 participants first just based on the definition and then again when they were confronted with infected[4] source code.

The main findings of our work are:

1. Some smells are considered important in theory but are not perceived as a design problem in practice.
2. Some smells are considered as not harmful in theory but are perceived as a design problem in practice.
3. Smells related to size and complexity are considered harmful by a higher percentage of participants than others.

The remainder of this paper is structured as follows. In Section 2, we discuss the background and related work: in Section 3, we describe the empirical study focusing on the study process, the study execution, and the data analysis. In Section 4, we present results obtained, while in Section 5 we discuss them. Section 6 discuss on threats to validity. Finally, in Section 7 we draw conclusions and outline future work.

2. Background and related work

In this section, we first introduce code smells and then report on empirical studies on them.

2.1. Code smells

Code smells, also referred to as bad smells, were first introduced by Kent Beck and Martin Fowler in 1999 [4], extending the code "pitfalls" defined in 1995 by Webster [28]. They can be considered as "poor" implementation and design decisions that may make it difficult for programmers to carry out changes and that hinder the evolution of systems. They are not considered defects but may increase the probability that flaws exist in a piece of software that may affect both the design stages and the implementation.

Table 1 presents the list of code smells proposed by Martin Fowler [4] and used in this study.

2.2. Empirical studies on code smells

Several studies found that code smells indeed are good indicators for code parts with a *low maintainability*, e.g., Deligiannis et al. [9] found that considering a specific design heuristic (to avoid a bad smell) in an experiment had a positive impact on creating more maintainable design structures; Malhotra et al. [10] found that bad smells could be used as an important source of information to *quantify flaws* in classes; and Fenske and Schulze [29] found that code smells that also consider variability, are good indicators of *low program comprehension, maintenance*, and *evolution* in software product lines.

Unfortunately, other studies found that considering code smells can have a negative impact on *maintainability* or that the impact is not always clear: e.g., Kim et al. [6] found that refactoring code clones did not always improve software quality; Yamashita and Moonen [7] found that only some code smells reflected important maintainability aspects, others required combining different

[1] https://marketplace.eclipse.org/content/jdeodorant.
[2] http://www.sonarqube.org/.
[3] Some authors instead of "harmfulness" speak of "criticality". Both terms express a potential risk attached to a problem but have slightly different connotations. For the sake of clarity, in this paper, we only use the term "harmfulness".
[4] We call source code "infected" if it contains code smells.

Please cite this article as: D. Taibi et al., How developers perceive smells in source code: A replicated study, Information and Software Technology (2017), http://dx.doi.org/10.1016/j.infsof.2017.08.008

Download English Version:

https://daneshyari.com/en/article/4972218

Download Persian Version:

https://daneshyari.com/article/4972218

Daneshyari.com