Università degli Studi di Salerno

Dipartimento di Scienze Aziendali – Management and Innovation Systems

---

Dottorato di Ricerca

Management and Information Technology

Curriculum in Informatica, Sistemi Informatici e Tecnologie del Software

XV Ciclo

Tesi di Dottorato

Code Smells: Relevance of the Problem and Novel Detection Techniques

Tutor

Prof. Andrea De Lucia

Coordinatore

Prof. Andrea De Lucia

Candidato

Fabio Palomba

Matr. 8887600001

Anno Accademico 2015/2016

Abstract

Software systems are becoming the core of the business of several industrial companies and, for this reason, they are getting bigger and more complex. Furthermore, they are subject of frantic modifications every day with regard to the implementation of new features or for bug fixing activities. In this context, often developers have not the possibility to design and implement ideal solutions, leading to the introduction of technical debt, *i.e.*, "not quite right code which we postpone making it right".

One noticeable symptom of technical debt is represented by the bad code smells, which were defined by Fowler to indicate sub-optimal design choices applied in the source code by developers. In the recent past, several studies have demonstrated the negative impact of code smells on the maintainability of the source code, as well as on the ability of developers to comprehend a software system. This is the reason why several automatic techniques and tools aimed at discovering portions of code affected by design flaws have been devised. Most of them rely on the analysis of the structural properties (*e.g.*, method calls) mined from the source code.

Despite the effort spent by the research community in recent years, there are still limitations that threat the industrial applicability of tools for detecting code smells. Specifically, there is a lack of evicence regarding (i) the circustamces leading to code smell introduction, (ii) the real impact of code smells on maintainability, since previous studies focused the attention on a limited number of software projects. Moreover, existing code smell detectors might be inadeguate for the detection of many code smells defined in literature. For instance, a number

of code smells are intrinsically characterized by how code elements change over time, rather than by structural properties extractable from the source code.

In the context of this thesis we face these specific challenges, by proposing a number of large-scale empirical investigations aimed at understanding (i) when and why smells are actually introduced, (ii) what is their longevity and the way developers remove them in practice, (iii) what is the impact of code smells on change- and fault-proneness, and (iv) how developers perceive code smells. At the same time, we devise two novel approaches for code smell detection that rely on alternative sources of information, *i.e.*, historical and textual, and we evaluate and compare their ability in detecting code smells with respect to other existing baseline approaches solely relying structural analysis.

The findings reported in this thesis somehow contradicts common expectations. In the first place, we demonstrate that code smells are usually introduced during the first commit on the repository involving a source file, and therefore they are not the result of frequent modifications during the history of source code. More importantly, almost 80% of the smells survive during the evolution, and the number of refactoring operations performed on them is dramatically low. Of these, only a small percentage actually removed a code smell. At the same time, we also found that code smells have a negative impact on maintainability, and in particular on both change- and fault-proneness of classes. In the second place, we demonstrate that developers can correctly perceive only a subset of code smells characterized by long or complex code, while the perception of other smells depend on the intensity with which they manifest themselves.

Furthermore, we also demonstrate the usefulness of historical and textual analysis as a way to improve existing detectors using orthogonal informations. The usage of these alternative sources of information help developers in correctly diagnose design problems and, therefore, they should be actively exploited in future research in the field.

Finally, we provide a set of open issues that need to be addressed by the research community in the future, as well as an overview of further future applications of code smells in other software engineering field.