A Systematic Literature Review on the Effect of Code Smells on Non-Functional Attributes of Source Code

Nazira Munalbayeva

March 26, 2020

1 Introduction

During the developing of a system, it's important to not only to simply "let the system works", but is also to produce a "good" system. A good system should follow the dependability metrics: a good system is mantainable: is easy to adapt to new change requests, is "readable" and can be modified easly. A good system is reliable: some systems will work in critical environments, so it's crucial that everything works as expected. A good system is available: having a system down, usually means loss of money, so a good system should be always available for users. A good system is secure: our system, and primarily IoT systems, usually collect sensible data, so it's our goal to keep all this information safe. Sadly, when we write code, we make errors, these errors are caused by many different reasons, like misunderstanding on specifications, communication issues, or simply, a tired programmer. In this project we will focus on Code Smells and non-functional attributes.

So, first we will define what is a non-functional attribute. As the name implies, non-functional requirements are not directly related to the functions performed by the system. They are associated with such integration properties of the system as fault tolerance, response time, usability, or system size. In addition, non-functional requirements may determine system limits, such as throughput of I / O devices, or data formats used in the system interface. Many non-functional requirements relate to the system as a whole, and not to its individual means. This means that they are more significant and critical than individual functional requirements. A mistake made in a functional requirement can reduce the quality of the system, an error in non-functional requirements can make the system inoperative. However, non-functional requirements may apply not only to the software system itself: some may relate to the technological process of creating software, others may contain a list of quality standards imposed on the development process. In addition, the specification of non-functional

requirements may indicate that the design of the system should be performed only by certain CASE-tools, and a description of the design process that must be followed.

2 Goals

In this project we will focus on Code Smells. A Code Smell it's not an error itself, it usually remarks some bad programming practice made by the developer or poor implementation and design choices made during the developing of the system. Some example of code smells can be: Feature envy, that denotes a strong coupling between classes, Dead Code, that denote a portion of code never used during the execution of the system, a Blob (or God Class) that is a long class containing dozens of methods, denoting a low cohesion of the class. As said these bad smells, are not properly an error, but they can lead to errors like faults, bugs or crash. Also someone can take advantage of this smells if the lead into a vulnerability. Or, simply, some code smells, like long methods, inconsistent variables name, poor/no comments, can make difficult to read the code, and this can affect the maintainability of the system.

Code smells, are studied for many reasons by the scientific community, first they are trying to understand how relevant smells are during the development of a system, how much the developers are aware about these smells and how much they are interested to avoid them. Also they are studying the impact of code smells on the overall quality and performance of the system. In the last years there have been also some attempt to develop some automatic tool that identify (and possibly correct) these smells. One example can be JDeodorant, an Eclpise plugin, developed and published by Concordia University that is able to identify some code smells like Feature Envy, Type Cheking, Long Method, God Class and Duplicated Code. But for other type of smells, or for more automatized processes the research is still going on.

With this project we want to investigate how, and how much these code smells impact on a system, and which are their effects on non-functional requirements. It is really worth to invest on research about code smell for a company? And again, mayor developer companies are really interested about code smells? In order to achieve our goal, we will review many scientific research and surveys in literature and try to answer to our questions.