Universidad Carlos III
Software Development 2023-24
Bachelor in Computer Science
Guided Exercise 2
Course 2023-24

GROUP: **88**

Emily Perez
100530320

Paper 1

Do We Have a Chance to Fix Bugs When Refactoring Code Smells? | IEEE Conference Publication | IEEE Xplore

This paper distinguishes the relationship between code smell detection and fault prediction which purpose is to prioritize code smell refactoring to reduce bugs. In hopes of doing so, they conduct studies on four open–source Java projects to reach answers to these questions:

- RQ1: is it possible to detect bugs during code smell refactoring?
- RQ2: to what extent can we improve the fault prediction performance using code smell detection results?
- RQ3: can we improve the efficiency of bug reduction during code smell refactoring?

It was found that some code smells can improve fault prediction precision and recall and that code smells have a correlation with fault prediction and it shows that there is a potential to detect and fix bugs during code smell refactoring. As well, they explore what code smells should be focused on based on fault prediction results and the result was that Blob, LongParameterList, and RefusedParentBeRequest should be prioritized to reduce bugs effectively.

The paper concludes by addressing potential threats to the validity of the study and proposing future research directions. These include further exploration of additional code smells and fault prediction models, as well as the investigation of strategies to effectively prioritize code smell refactoring.

Paper 2

Code Smells and Detection Techniques: A Survey | IEEE Conference Publication | IEEE Xplore

Code smells are signs of potential issues affecting software quality and maintainability. Refactoring goals are to eliminate these code smells without altering the product of the code. The study aims to support other research in regards to exploring code smells, detection approaches, and other commonly used tools.
There were 19 previous studies that were analyzed to identify and discuss code smells, detection approaches, and refactoring tools. In this paper, it is noted that the most important code smells are duplicated code, long method, and accidental complexity. There are various ways to detect code smells like manual,

metrics-based, history-based, machine learning-based, software visualization, and semi-automated tools. It is important to note that the accuracy of such tools vary.

In conclusion, The study highlights limitations and challenges in the field, including the dominance of tools supporting the Java language, inconsistency in threshold values for metrics-based approaches, and the performance dependency of machine learning algorithms on training datasets.