SSW567

Impact of Code Duplicates Removal on Quality - Phase I

Group 6: Jiayin Huang, Sijie Yu, Rongda Kan, Yulong Yan

As business applications and software have become more complex and feature-rich, code bases become larger and the resources that are required to run them have grown exponentially over the last two decades. Code quality is important for ensuring reliability and scalability for the change. A great interest has been observed in the literature in the identification of parts of the code that need to be improved. In particular, duplicated code is one of the most pervasive and pungent smells to remove from source code through refactoring. It is common knowledge that code duplicates could have a number of negative impacts on a software system. Such as the potential of running into bugs, large code size, maintenance costs, readability, and usefulness, etc. Existing studies present multiple methodologies to identify and remove code duplicates. However, more quantitative studies are needed to measure the severity of the impact on systems. This would help stakeholders to identify the importance of improving code quality and funnel more research resources to develop better applications that help to remove code duplicates.

We extensively brief through the literature on quality attributes. Duplicated code has been largely studied, and different types of duplications have been identified. Georges divides these duplicated codes into two categories that a complete function is duplicated exactly or only a piece of code is found as part of a number of functions. And he concludes that by eliminating the duplicates, you ensure that the code says everything once and only once, which is a rule of good removal[1]. Some studies have experimented with different refactoring tools to remove code duplication. The refactoring tool that they found more useful during the experimentation is Eclipse and it's very simple and intuitive to use[2]. Another useful approach and tool developed to suggest code duplicate removal is called Duplicated Code Refactoring Advisor (DCRA) and is composed of four modules[3].

As we continue to investigate existing studies and complete preliminary research, we may identify other perspectives for validating impact of code duplicates.

It will be essential to use a systematic approach that takes into account both the technical and non-technical facets of software development in order to address the effect of refactoring on the relationship between quality attributes and design metrics.
Under the above circumstances, this paper aims to conduct research on code refactoring with underlying variables. More specifically, our goal is to analyze data generated from code for understanding whether refactoring duplicates could impact code quality and performance by applying methodologies:

1) Data preparation. Sample should be cleaned up and represents the majority of SDLC processes.
2) Identifying and measuring the size of duplicates. We will leverage existing methods, set up critical quantified metrics that indicate code duplicates in the system.
3) Measuring performance under a controlled experiment. We will compare performance of datasets of both before and after code refactoring. We will continue to investigate the right methods to analyze and measure the severity impact.

[1] Georges G. K. : A Scenario Based Approach for Refactoring Duplicated Code in Object Oriented Systems.
[2] Francesca A. F., Marco M., Domenico P., Marco Z.: On Experimenting Refactoring Tools to Remove Code Smells. XP'15 workshops: Scientific Workshop PRoceedings of the XP2015.
[3] Francesca A. F., Marco Z., Francesco Z.: A Duplicated Code Refactoring Advisor. Agile Processes in Software Engineering and Extreme Programming pp 3-14.