Software Development GE4

Guillermo Villar Sánchez (100472121)
Alberto Sánchez del Álamo (100472120)

Making Program Refactoring Safer

https://dl.acm.org/doi/pdf/10.1145/1810295.1810461?casa_token=-Wpts92jWUkAAAAA:4Hewj41QRr3bXm-svQU7bu_HCFbfGfrMaqKHa-T0lYwPxVh09kxFkHNzU_B8SSp3PWoimndoFZYMUQ

When dealing with refactoring developers usually use tools integrated in IDEs such as Eclipse, NetBeans, JBuilder, and IntelliJ, which automate the process. However, IDEs may perform non-behavior-preserving transformations, usually when applying the pull up refactoring.

SafeRefactor is a tool integrated into the Eclipse IDE which checks the refactoring in sequential Java programs. This plugin generates and a run a bank of tests for the methods refactored, in order to check if the behavior after the refactoring test is the same as before. Finally, a report with the results is showed to the developer indicating the correct and wrong methods.

Overall, *SafeRefactor* can be a very useful tool to check the correct functioning of the refactored code and to identify new bugs.

A Field Study of Refactoring Challenges and Benefits

https://dl.acm.org/doi/abs/10.1145/2393596.2393655

https://dl.acm.org/doi/pdf/10.1145/2393596.2393655?casa_token=l7KDg-DDw5IAAAAA:GKyVqkL89ZYr6KOHaUlPqAg5AblDTx1Z4pwKJ3fVlfT2ixVOF08tFh18Kla_M8SOMU26rECppiS-Vg

The paper is a systematic review (meaning its not specifically a paper but more of a general review of literature on the topic) of empirically (following scientific standards) evaluated techniques for identifying opportunities for refactoring object-oriented code. The study found that identifying refactoring opportunities is a highly active area of research, with Move Method, Extract Class, and Extract Method being the most frequently studied refactoring activities.

The paper identifies six main approaches followed by researchers to identify refactoring opportunities, and notes that most researchers use well-known software engineering-related empirical evaluation approaches. The study recommends that researchers invite industry experts to participate, expand the coverage of their work to include more refactoring activities, use relatively large data sets implemented with different programming languages, and pay greater attention to certain research points to increase the confidence in their empirical evaluation results.

As a general conclusion, the paper finds refactoring a sector of the industry where there is a lot of development and opportunities, as it's a less explored field.

The impact of Refactoring on Quality and Productivity: An Empirical study

https://www.matec-conferences.org/articles/matecconf/pdf/2016/20/matecconf_icaet2016_02012.pdf

The paper "The Impact of Refactoring on Quality and Productivity: An Empirical Study" talks about how code refactoring can be used to improve the quality of software programs. Refactoring involves changing the structure of code without changing its functionality. It can help make the code easier to understand, maintain, and modify, and can also improve its performance.

The paper presents findings from a study of four open-source software projects that used refactoring techniques. The researchers found that refactoring improved the overall quality of the code, as well as making it easier to maintain and modify. They also found that refactoring can help identify and fix issues with code that might not have been caught otherwise.

The study identified some challenges to refactoring, including the time and effort required to perform it and the need for coordination between team members. However, the benefits of refactoring were seen to outweigh these challenges.

Overall, the paper suggests that code refactoring is an effective way to improve software quality and should be considered as a regular practice in software development.

Evaluation of the impact of code refactoring on embedded software efficiency

https://www.researchgate.net/profile/Lisane-Brisolara/publication/229519223_Evaluation_of_the_impact_of_code_refactoring_on_embedded_software_efficiency/links/0046353cbf08537cc8000000/Evaluation-of-the-impact-of-code-refactoring-on-embedded-software-efficiency.pdf

This paper looks at how using object-oriented languages for embedded software can negatively affect energy consumption and performance. To improve software quality, code refactoring techniques are used. This paper focuses on one specific refactoring technique called inline method refactoring and analyzes how it impacts the performance and energy consumption of embedded software written in Java.

The experiment was conducted using three different applications and an estimation tool called DESEJOS on the FemtoJava processor. The results show that the inline method refactoring can improve performance and decrease energy consumption in simpler applications but not in more complex ones.

The paper concludes that when applying inline method refactoring, the complexity of the method and the whole application should be taken into account. Future work will explore other refactoring methods and case studies.