ABOUT REFACTORING: A BRIEF ARTICLE

Kaur, Amandeep. Kaur, Manpreet. Analysis of Code Refactoring Impact on Software Quality. EDP Sciences. May 11, 2016.

The refactoring technique is based on changing the internal structure of a software in order to make it easier to understand, while not changing the functionality of it. Some advantages of this methodology include a reduction of time for code maintenance and improved code readability. Although manual refactoring may cause different problems such as error-prone and time consuming, automatic refactoring reduces these problems.

Vidal, Santiago A.. Claudia Marcos. Díaz-Pace, J. Andrés. An approach to prioritize code smells for refactoring. Springer Link. December 14, 2014.

The number of code smells, which are the equivalent of “fragments of code to refactor”, sometimes may exceed the capabilities of the developer for fixing them. Furthermore, not all code smells are as important as others. For it, this article describes a ranking of code smells priority. However, this prioritization may be subjective and dependent for each project, as every software has a different architecture model with different oriented languages.

Paixão, Matheus. Uchôa, Anderson. Bibiano, Ana Carla. Oliveira, Daniel. Garcia, Alessandro. Krinke, Jens. Arvonio, Emilio. Behind the Intents: An In-depth Empirical Study on Software Refactoring in Modern Code Review. MSR ‘20. October 5-6, 2020.

This text stresses the significance of both simple and complex refactorings in feature-addition code reviews. It notes that while explicit refactoring changes often involve complex operations, they undergo fewer revisions and are rarely refined during reviews. Additionally, it highlights how new developers' intentions commonly influence the selection and composition of refactorings, underscoring the need for refining techniques for refactoring-aware code reviews.

Keuning, Hieke. Heeren, Bastiaan. Jeuring, Johan. A Tutoring System to Learn Code Refactoring. SIGCSE ‘21. March 13-20, 2021.

It is known that a lot of code errors made by students are due to misconceptions and bad understanding of code. Furthermore, not all solutions proposed are equally good, as several factors must be considered. Recently, educators have been taking more into account readability and style quality of the code. Moreover, professors usually try their best correcting the student’s code, but due to large classes, code duplication or long code extension, it is very hard to propose an individual solution for each code. That is why the use of refactoring is essential for the modern programmer.

Sagar, Priyadarshni Suresh. AlOmar, Eman Abdulah. Mkaouer, Mohamed Wiem. Ouni, Ali. Newman, Christian D. Comparing Commit Messages and Source Code Metrics for the Prediction Refactoring Activities. MDPI. September 30, 2021.

It is commonly known that a commit in a historial version (for instance *Github*), must be descriptive enough for other developers to understand what has been done. This must be applied to refactoring techniques also. The usage of commit messages such as *Move Method, Extract Method, Rename method*,... are fundamental for time optimization and understanding what other developers may have done to the code.

Oliveira Naelson. Ribeiro, Márcio. Bonifácio, Rodrigo. Gheyi, Rohit. Wiese, Igor. Fonseca, Baldoino. Lint-Based Warnings in Python Code: Frequency, Awareness and Refactoring. IEEE. January 12, 2023.

The linters are tools used for following a declared notation on the code. Surprisingly, it was discovered that nearly 40% of the projects in Python to study had at least one Pylint warning. It was also found that almost 70% of the developers prefer Python code without lint-based warnings. The study followed more than fifty developers and six common Pylint warnings in the pull-requests of the project's repositories.

.