Refactoring - Analysis on Articles

Alejandro Sanchez Vega - 100495744
Iciar García Izquierdo - 100495789
Isabel Hernanz García - 100495778

Article 1 - Software Refactoring Prediction Using SVM and Optimization Algorithms

Publisher: MDPI - Challenges in Machine Learning, Artificial Intelligence, Wireless Sensor Networks and Smart Cities
Published: 15 August 2022
Address: https://www.mdpi.com/2227-9717/10/8/1611
Authors: Mohammed Akour, Mamdouh Alenezi, Hiba Alsghaier

When thinking about solving quality issues and code smells, refactoring is the first idea that comes to our minds but, it still is a challenge for many software developers. This article presents a contribution to predicting refactoring needs at a class-level, with the open source systems ANTLR4, JUnit, MapDB, and McMMO, and the usage of two optimization algorithms: Genetic and Whale algorithms. This prediction on refactoring uses a vector based machine that will go through 4 steps: A pre-processing phase with already collected datasets, an application of the optimization algorithms and SVM as classificators on the processed datasets, an evaluation of results with a signed rank test and at last a comparison of results.

The genetic optimization algorithm is based on heuristic search, and standard search too. It provides approaches to search problems based on a natural selection process, and so it is very common for software fault prediction. On the other hand, the whale algorithm is based on the way of life of whales, searching for the best solution on the solutions' population.
This article assures the effectiveness of machine learning algorithms on refactoring predictions, but considers that the two optimization algorithms used for this study were used only for that time at class level, so the accuracy level was not that high. However, the project revealed a rank of accuracy of 84% for one of the systems, and 93% for the other one.

Article 2 - Toward Understanding the Impact of Refactoring on Program Comprehension

Publisher: Institute of Electrical and Electronics Engineers (IEEE)
Published: March 2022
Address:
https://ieeexplore.ieee.org/abstract/document/9825885?casa_token=Ask7FfvjSgQAAAAA:p0wHFvj4xPeK9RUBvsV3T29p_E2txQwzTTks4yvEwVuNbfAGFojW58ufGvg8PwNgLC7r-EsoEg
Authors: Giulia Sellitto, Emanuele Iannone, Zadia Codabux, Valentina Lenarduzzi, Andrea De Lucia, Fabio Palomba, Filomena Ferrucci.

Software refactoring is known to be useful to improve maintainability and efficiency on our code, as well as helping coders to understand it better. This article studies around the basic measure of coding comprehension, which is known as *program readability*, in a sample of 156 projects and, by mining the refactoring data from those samples, it shows a quantification of the refactoring done in them. The methodology used is based on studying the history of all sample projects and identifying the commits where refactoring has been

applied once or at more occasions. The next followed step is computing the readability metrics defined by the Scalabrino model.

After the research, an analysis was done, identifying a relationship between the refactorings that have been applied to the source codes, and the variations in their program readability. The strategy was simple: If the variation on the readability was over 0, the code was categorized as "increased", and if it was below 0, it was moved to the "decreased" category. Then, eight statistical models were designed in R, and the study revealed that not always the refactoring was indeed useful to improve the understandability of the code. Anyhow, it was important to signal that this depended on the aim of each refactor technique, since those targeted to increase cohesion generally did improve the readability of the source code.

Article 3 - Refactoring – Does it improve software quality?
Publisher: IEEE
Published: May 2007
Address: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4273477
Authors: Konstantinos Stroggylos, Diomidis Spinellis

During this study the authors try to corroborate the idea that refactoring does improve the quality of the code that is being refactored. For this purpose they use some metrics defined by previous scientists to see the relation between refactoring and the improvements or change in those metrics. They did also develop some tools to help the recording and change of these metrics, they used C, SQL and Java for this. At the end of the article they present the different metrics they got and some Statistical measures, the ending result was that refactoring changed to the worse some metrics observed , but it was attached as a consequence of the bases of the metrics observed. Even Though the results were somewhat negative, they close the article expressing their expectations for future studies and the possibility to correlate the different refactoring methods to the change of some metrics.

Article 4 - Analysis of Code Refactoring Impact on Software Quality
Publisher: Matec Web of conferences
Published : May 2016
Address:
https://www.matec-conferences.org/articles/matecconf/pdf/2016/20/matecconf_icaet2016_02012.pdf
Authors: Amandeep Kaur1 and Manpreet Kaur2

This article works as a great introduction to refactoring. It is lightworded and explains in a simple manner what refactoring is , some concepts regarding refactoring, and a tool and techniques to implement refactoring in our code. I could find some of the concepts seen in class explained in the article as Bad Smells and their types, and techniques of refactoring.
An interesting fact of the article is the explanation of a method for finding bad smells and the refactor of them, with the use of a flow diagram, after that they expose an experiment done to

express the difference between a code with refactoring and another no refactored. The conclusion they got was that refactoring does improve the readability of the code and in some cases their performance.

Article 5 - Green software: Refactoring approach
Publisher: Journal of King Saud University - Computer and Information Sciences, Elsevier
Published: July 2022
Address: https://www.sciencedirect.com/science/article/pii/S1319157820305164
Authors: Rajni Sehgal, Deepti Mehrotra, Renuka Nagpal, Ramanuj Sharma

This article will explore the different refactoring techniques and their consequences regarding energy consumption in mobile computing applications developed in Java. To validate this, a statistical T test is performed in order to observe such differences (before and after refactoring of the code). The ultimate goal of the study is to find out if the application of these techniques can improve the efficiency of the code or it can reduce the energy leak that happens with code smells. It was discovered that there was an impact on battery usage with the presence of code smells. It was concluded the reduction in power usage is deeply correlated with refactoring. Also, the code quality was severely improven.

Article 6 - Industry experiences with large-scale refactoring
Publisher: Association for Computing Machinery, New York, NY, United States
Published: 09 November 2022
Address: https://doi.org/10.1145/3540250.3558954
Authors: James Ivers, Robert L. Nord, Ipek Ozkaya, Chris Seifried, Christopher S. Timperley, and Marouane Kessentini.
This last article explains the differences between refactoring and large-scale refactoring, taking this last one a bigger amount of time to develop successfully. In order to capture this challenge they did a survey with developers with experience. Even thought, they used well known techniques in order to carry out this job, but still faced multiple challenges. Making clear that new refactoring techniques were needed. However, many developers achieved a high percentage of their goals.