Article 1:
"Green software: Refactoring approach" by Rajni Sehgal, Deepti Mehrotra, Renuka Nagpal, and Ramanuj Sharma at Amity University Uttar Pradesh, India
https://www.sciencedirect.com/science/article/pii/S1319157820305164
Date of Publication (most recent revision): June 28, 2022
Summary:
With many software programs there are energy leaks due to useless processes or unused code that is being run constantly and wasting energy. These are accompanied with code smells that are introduced in the development phase, and many times increase energy leaks. These energy leaks and code smells are addressed during the refactoring process. This publication is looking into the effect of refactoring on energy consumption. This is done by looking at multiple programs before and after refactoring and looking at the energy usage of each. From this it was found that having code smells causes much higher energy and battery usage. It was also found that refactoring was an effective way to have better power saving and reduced energy consumption as well as reducing code smells.

Article 2:
"Behind the scenes: On the relationship between developer experience and refactoring" by Eman Abdullah Alomar, Anthony Peruma, Mohamed Wiem Mkaouer, Christian D. Newman, Ali Ouni
https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2395
Publication Date: 28 January 2024
Summary:
Past refactoring surveys have shown that code refactoring is mainly done by developers who have extended knowledge of the system's design and have some kind of leadership role in their team. These surveys were limited in scope, however, and this study seeks to determine the accuracy of these previous studies through an analysis of 800 open-source projects. The study analyzed several refactoring activities, and assigned an experience score to each developer taking part. The research seeks to determine if developers with higher scores tend to perform more code refactorings, if different developers expressed different motivations behind their refactorings, and if people across different scores documented their refactoring activity differently. In conclusion, the study found that those with higher scores tended to perform more refactorings, that there seemed to be no correlation between experience level and motivation behind refactoring, although those with higher scores tended to perform a higher range of refactoring activities. Finally, it found that more experienced developers tended to document their refactoring less. Overall, we found that those in charge of the majority of refactoring activities are those in advanced positions in development teams.

Article 3: "A Refactoring Classification Framework for Efficient Software Maintenance" by Abdullah Almogahed, Hairulnizam Mahdidi Omar, Nur Haryani Zakaria, Salama A. Mostafa
https://ieeexplore.ieee.org/abstract/document/10192906/authors

Date of Publication: 25 July 2023

Regarding total costs of software development, the process of maintenance and evolution consists of about 80% of the overall costs. Refactoring plays a large role in streamlining the maintenance process, however the processes and effects of refactoring are still very inconsistent. This study attempts to create a framework for classifying refactoring strategies based on the influence and efficiency. First, an exploratory study was conducted to identify refactoring techniques. Next, a series of five case studies were conducted to note the most important attributes of refactoring. The proposed framework consists of the methodology for applying refactoring techniques, the Quality Model for Object-Oriented Design, and the classification scheme for refactoring techniques. Developers can use this framework to decide which refactoring technique will best suit their needs.

Article 4:

"Fixing Your Own Smells: Adding a Mistake-Based Familiarisation Step When Teaching Code Refactoring" by Ivan Tan and Christopher M. Poskitt at Singapore Management University

https://dl.acm.org/doi/epdf/10.1145/3626252.3630856

Publication Date: March 7th, 2024

Summary:

This publication focuses on how students should be taught about refactoring code in college as an undergraduate to make them learn it more effectively and be able to apply it more effectively. This study does this by using a mistake based approach in which it first gives a project to students which is written in a way that the students will create with code smells. They were then taught about these code smells and told to go back through their own code to refactor it. This is showing how students are better able to refactor and learn refactoring when working with a product they created. This study suggests that by teaching refactoring this way it will help students better recognize code smells as they are developing their program to overall produce a better product the first time around and having less to refactor after the fact.

Article 5

"How to Refactor this Code? An Exploratory Study on Developer-ChatGPT Refactoring Conversations" by Eman Abdullah AlOmar, Anushkrishna Venkatakrishnan, Mohamed Wiem Mkaouer, Christian D. Newman, and Ali Ouni on Arxiv.

https://doi.org/10.48550/arXiv.2402.06013

Publication Date: February 8th, 2024

This article explains the importance of understanding how developers use Large Language Models (LLMs), like ChatGPT, specifically for code refactoring. The paper investigates the

interaction between developers and ChatGPT, focusing on how developers communicate their needs for refactoring and how the model assists them. By analyzing thousands of these interactions, the study identifies common ways developers ask for help and how ChatGPT responds, providing insights into how AI can better support code refactoring tasks. This research is significant because it helps improve the tools developers use to make coding more efficient and less error-prone, which is crucial in building reliable and maintainable software.

Article 6

"Automating Source Code Refactoring in the Classroom" by Eman Abdullah AlOmar, Mohamed Wiem Mkaouer, and Ali Ouni

SIGCSE 2024: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V.1; Pages 60–66
https://doi.org/10.1145/3626252.3630787
Publication Date: March, 2024

This article highlights the significance of teaching code refactoring in educational settings, specifically how it can help students identify and correct poor coding practices known as antipatterns. The paper discusses an experiment where students used a tool called JDeodorant, an IDE plugin designed to detect antipatterns and assist in refactoring, within a classroom setting. The study found that students not only learned to appreciate the value of refactoring but also gained satisfaction from using the JDeodorant tool, which helped them improve their coding skills by removing inefficient or problematic patterns. By sharing these findings, the paper supports the idea that integrating refactoring tools and practices into computing education can prepare students to write cleaner, more efficient code, making them better programmers and enhancing the reliability of the software they will develop in the future.