Software Development

DEGREE IN COMPUTER SCIENCE & ENGINEERING

Group Members:

Carlos Martin Gallardo - 100522258@alumnos.uc3m.com

Alejandro Quirante Sanz - 100522183@alumnos.uc3m.com

Group 88 - Team 05 - Computer Science & Engineering

Exercise 3

Refactoring and Simple Design

Academic Year 2024/2025

Article 1:

“How to refactor this code? An exploratory study on developer-ChatGPT refactoring conversations”

(https://doi.org/10.1145/3643991.364508)

Conference: MSR '24: Proceedings of the 21st International Conference on Mining Software Repositories

Authors: Eman Abdullah AlOmar, Anushkrishna Venkatakrishnan, Mohamed Wiem Mkaouer, Christian Newman & Ali Ouni

Publication Date: 02 July 2024

Summary:

Large Language Models (LLMs) like ChatGPT are increasingly being used in software engineering tasks such as code refactoring, testing, reviewing, and understanding. While past research has examined how refactoring is documented in areas like commit messages and code reviews, there's limited insight into how developers express their refactoring needs when using ChatGPT. This study aims to fill that gap by analyzing developer-ChatGPT conversations to understand how developers recognize and communicate the need for code improvements, and how ChatGPT responds. By examining nearly 18,000 prompts and responses, the study finds that developers tend to use both broad and specific language when discussing refactoring, often provide vague requests, and rely on ChatGPT to clarify intent. The analysis also uncovers different ways developers frame their prompts depending on what they want to learn. These insights shed light on how developers and AI tools collaborate in the refactoring process.

Article 2:

“Semantic Code Refactoring for Abstract Data Types”

(https://doi.org/10.1145/3632870)

Journal: Proceedings of the ACM on Programming Languages, Volume 8, Issue POPL

Authors: Shankara Pailoor, Yuepeng Wang & Işıl Dillig

Publication Date: 05 january 2024

Summary:

Changes to how data is represented in an abstract data type (ADT) often demand deep semantic code refactoring, which can be complex and time-consuming. To address this, the paper introduces an automated method for semantic refactoring. Given an original ADT, a new data structure, and a relational invariant connecting the two, the method generates a semantically equivalent ADT implementation. Built on counterexample-guided inductive synthesis (CEGIS), the approach incorporates three key innovations: breaking down the problem into simpler programming-by-example tasks for each method, using logical abduction to infer useful code patterns, and applying partial equivalence to improve synthesis effectiveness. The technique is implemented in a tool called Revamp, which was tested on 30 Java classes from GitHub. The results show high accuracy, with Revamp successfully refactoring 97% of the ADTs and re-implementing nearly all modified methods.

Article 3:

“Refactoring Impact Prediction for Code Quality Metrics”
(https://ieeexplore.ieee.org/abstract/document/6976117)

Journal: IEEE explore

Authors: Chao Liu, Wei Tong, Hadi D. Ochoa & Radu Prodan
Publication Date: 01 April 2024

Summary:

The paper discusses the development of RIPE, a tool designed to predict the impact of 12 common refactoring techniques on 11 key code quality metrics, such as coupling, cohesion, complexity, and size. While refactoring is intended to improve code quality, it can inadvertently introduce bugs and remove necessary structures. RIPE employs heuristic-based prediction functions and was empirically tested on 504 refactorings from 15 Java open-source systems, achieving a 38% accuracy in predicting 8,103 metric values, with better performance on simpler refactorings like Pull Up Method compared to more complex ones like Extract Class. The tool aids developers in assessing the potential effects of refactorings, especially when conflicting changes arise, but the authors acknowledge the need for improved accuracy in predicting complex refactorings. Future enhancements for RIPE include expanding its range of supported refactorings and metrics, as well as enabling predictions for sequences of refactorings to provide a more comprehensive understanding of code restructuring.

Article 4:

"A Survey on Secure Refactoring"

(https://link.springer.com/article/10.1007/s42979-024-03325-y)

Journal: Springer Nature Link

Authors: Estomii Edward, Ally S. Nyamawe, Noe Elisa

Publication Date: 20 September 2024

Summary:

The article explores the concept of secure refactoring in software development, which involves restructuring code to enhance design, maintainability, and readability while addressing security vulnerabilities. By reviewing 55 studies from 2011 to 2023, the authors identify various refactoring techniques, such as Extract Method and Encapsulate Field, that can mitigate common security risks like SQL Injection and Cross-Site Scripting. A key contribution of the study is the development of a taxonomy for secure refactoring methods, which categorizes approaches to tackle different security issues. The authors note the limited research on how refactoring might introduce new vulnerabilities and emphasize the need to integrate security into the refactoring process. They advocate for more automated solutions to assist developers in consistently addressing security during code restructuring, ultimately highlighting the importance of considering security in refactoring to enhance the overall safety of software systems.