Software Development
Course 2024-25

# Guided Exercise 3

Date: **22/04/25**

Authors:
**JAANAVI THANAMALA - 100559609**
**ARYAN VERMA - 100465924**

## INDEX

## Articles

Jaanavi Thanamala

1. **Behind the intents: An in-depth empirical study on software refactoring in modern code review**
   This study investigates how software refactoring is proposed, discussed, and justified during modern code reviews. By analyzing a large set of code review discussions from open-source projects, the authors identify common intentions behind refactoring requests—such as improving readability, maintainability, or performance. The research highlights the importance of communication and context in refactoring decisions, showing that reviewers often request refactorings to align code with best practices and team standards.

   Paixao, M., Uchôa, A., Bibiano, A. C., Oliveira, D., Garcia, A., Krinke, J., & Arnovio, E. (2020, October). Behind the intents: An in-depth empirical study on software refactoring in modern code review. In Proceedings of the 17th IEEE/ACM International Conference on Mining Software Repositories (MSR 2020) (pp. 456–467). IEEE.

2. **An interactive and dynamic search-based approach to software refactoring recommendations**
   This paper introduces a novel approach that leverages search-based software engineering to recommend refactoring opportunities. The method uses multi-objective optimization to balance different quality metrics (like code complexity and coupling) and incorporates developer feedback to refine suggestions. The interactive system dynamically adapts recommendations based on user input, making the refactoring process more effective and tailored to project needs.

   Alizadeh, V., Kessentini, M., Mkaouer, M. W., O'Cinneide, M., Ouni, A., & Cai, Y. (2020). An interactive and dynamic search-based approach to software refactoring recommendations. IEEE Transactions on Software Engineering, 46(9), 932–961.

Aryan Verma

1. **Usability improvement through A/B testing and refactoring**
   The authors combine A/B testing with systematic refactoring to enhance the usability of web interfaces. By deploying alternative interface versions and analyzing user behavior, they identify which design changes improve user experience. Subsequent refactoring then streamlines the codebase to integrate these improvements, demonstrating that coupling empirical user feedback with code restructuring leads to measurable gains in interface quality and maintainability.

   Sergio Firmenich, Alejandra Garrido, Julián Grigera, José Matías Rivero, and Gustavo Rossi. 2019. Usability improvement through A/B testing and refactoring. Software Quality Journal 27, 1 (March 2019), 203–240.

2. **An empirical study to improve software security through the application of code refactoring**
This empirical study examines the impact of code refactoring on software security. By applying refactoring techniques to real-world codebases and measuring security metrics before and after, the authors find that targeted refactoring can significantly reduce vulnerabilities. The research provides evidence that improving code structure not only enhances maintainability but also strengthens the overall security posture of software systems.

   Abdullah Almogahed, Mazni Omar, and Nur Haryani Zakaria. 2022. Refactoring Codes to Improve Software Security Requirements. Procedia Comput. Sci. 204, C (2022), 108–115.