



Ph.D. DISSERTATION DEFENSE

Candidate:	Gengwu Zhao
Degree:	Doctor of Philosophy
School/Department:	Charles V. Schaefer, Jr. School of Engineering and Science Department of Systems Engineering
Date:	Tuesday, November 18th, 2025
Time/Location:	from 1:00 to 3:00 PM. Babbio 503
Title:	Detecting and Refactoring Mock Clones in Java Test Code: From Empirical Evidence to Automated Framework
Chairperson:	Dr. Lu Xiao, Charles V. Schaefer, Jr. School of Engineering and Science, Department of Systems Engineering
Committee Members:	Dr. Eman Alomar, Charles V. Schaefer, Jr. School of Engineering and Science, Department of Systems Engineering Dr. Ying Wang, Charles V. Schaefer, Jr. School of Engineering and Science, Department of Systems Engineering Dr. Sang Won Bae, Charles V. Schaefer, Jr. School of Engineering and Science, Department of Systems Engineering Dr. Tegan Brennan, Charles V. Schaefer, Jr. School of Engineering and Science, Department of Computer Science

ABSTRACT

Mocking frameworks, such as Mockito, EasyMock, and PowerMock, have become indispensable in modern software testing by enabling developers to isolate test dependencies and improve test controllability. However, as mocking becomes increasingly common, it introduces a distinctive form of redundancy that differs fundamentally from traditional code clones. Traditional clones typically involve contiguous blocks of syntactically similar code, whereas mock clones consist of repeated and semantically similar API sequences—mock creation, stubbing, and verification—scattered across different test cases. These sequences are fine-grained and often interleaved with unrelated test logic, making them invisible to conventional clone detection tools that rely on structural or textual similarity. As a result, existing clone detectors fail to capture these subtle yet pervasive redundancies in test suites. Identifying and refactoring mock clones can substantially improve test readability, enhance maintainability, and reduce code complexity, ultimately leading to more robust and evolvable testing practices.

The research unfolds in three phases. First, an empirical study of 193 Apache Java projects reveals that mocking frameworks are widely adopted in 66% of projects, with Mockito being dominant. Despite this popularity, mocking is practiced selectively, and developers tend to reuse a small set of APIs repeatedly. The



top five mocking APIs account for over 78--100% of usage, suggesting the presence of structural redundancy in test code.

Building upon these insights, the second phase conducts a comprehensive investigation of mock clones across six large open-source projects. The study identifies 6,547 mock objects and 1,363 frequently mocked classes, uncovering 698 mock clone instances that affect 87% of frequently mocked classes. Existing code clone detection tools fail to capture these fine-grained, interleaved, and semantically varied mocking patterns. Manual refactoring demonstrates that eliminating mock clones can reduce up to 61% of mocking-related lines of code while preserving behavior.

Finally, the third phase introduces CloneDeMocker, the first automated framework for detecting and refactoring mock clones. Detection is formulated as an optimization problem, combining Apriori-based frequent stub-set mining and greedy clustering. Refactoring is guided by large language models (LLMs) via a modular, template-driven pipeline. Evaluations show that CloneDeMocker achieves 89% refactoring success across 86% of affected test cases, reducing complexity (CCTR) by 10--40% on average, with 20% exceeding 50%, at an overall cost of only \$14.41.

This dissertation formalizes mock clones as a distinct form of fine-grained duplication in unit testing, empirically establishes their prevalence and impact, and provides an end-to-end automated solution for their detection and refactoring. The resulting framework and datasets lay the foundation for future research on mock-aware program analysis, clone evolution, and intelligent test refactoring.