# Improving Software Maintainability through Automated Refactoring of Code Clones

1st Simon Baars
*University of Amsterdam*
Amsterdam, the Netherlands
simon.mailadres@gmail.com

2nd Ana Oprescu
*University of Amsterdam*
Amsterdam, the Netherlands
ana.oprescu@uva.nl

***Abstract*—Coming up!**

***Index Terms*—code clones, refactoring, static code analysis, object-oriented programming**

## I. Introduction

## II. Background

### A. Code clone terminology

Clone class collection, clone class, clone instance, cloned node, token.

### B. Code clone definitions

Quick overview of clone type definitions

### C. Refactoring techniques

Extract method, move method.

## III. Defining refactorable clones

Instead of defining clone types, I just define how we can ensure that clones can be refactored. I do not plan to differentiate between T1R, T2R and T3R here, because I think it makes the paper confusing.

To be honest, I'm not even sure how relevant it is to name the literature clone types. Maybe not focus on their shortcomings, but just start from the core (sourcecode) and define how we can get to refactorable clones.

### A. Ensuring Equality

Explain type 1, why it is not always refactorable, solution

### B. Allowing variability in a controlled set of expressions

Explain type 2, why it is not always refactorable, solution

### C. Gapped clones

Explain type 3, why it is not always refactorable, solution

## IV. CloneRefactor

The tool: Detect Clones, Map Context, Refactor.

### A. Clone Detection

Detecting refactorable clones, (clone graph?? do I want to explain my exact methods?)

### B. Context Mapping

*1) Relation:* The rationale for our categories regarding clone relations.

*2) Location:* The rationale for our categories regarding clone locations.

*3) Contents:* The rationale for our categories regarding clone contents.

### C. Refactoring

*1) Extract Method:* Show my categories to show what clones can be dealt with by method extraction.

*2) AST Transformation:* Explain what AST transformations I do to apply the refactorings.

*3) The cyclic nature of refactoring:* Explain how refactoring code clones might open up new refactoring opportunities. We refactor a project until there are no more open refactoring opportunities.

## V. Results

### A. Clone context

How many clones are there in certain contexts? Experiments for relation, location and context.

### B. Clone refactorability

To what extent can found clones be refactored through method extraction, without requiring additional transformations.

### C. Thresholds

I think the ultimate goal with this thesis is to do experiments with different clone thresholds. Which thresholds give clones that we should refactor? For this, we will measure the maintainability of the refactored source code over different thresholds. These thresholds range from minimum clone size, variability and gap size.

## VI. Discussion

### A. Clone Definitions

### B. CloneRefactor

### C. Experimental setup

## VII. Conclusion

## Acknowledgment

:-)

## References