# Advanced Clone-Analysis to Support Object-Oriented System Refactoring

Magdalena Balazinska[1], Ettore Merlo[2], Michel Dagenais[2], Bruno Lague[3], Kostas Kontogiannis[4]

(1) Dept. of Computer Science and Engineering, University of Washington,
(2) Dept. of Computer and Software Engineering, École Polytechnique de Montréal,
(3) BroadSoft, Montreal, (4) Dept. of Electrical and Computer Engineering, University of Waterloo

## 1. Abstract

Clone detection and re-factoring have grown in importance over the past 10 years. In this talk, we will briefly review the WCRE 2000 work and discuss the advances in the field.

The WCRE 2000 paper presented a computer assisted clone re-factoring approach. The process was based on metric-based clone analysis that produced clone clusters. Clones in the same cluster were then compared using token-based dynamic programming (DP) matching. Token-based clone differences, which included insertions, deletions, and substitutions, were then projected on to the ASTs corresponding to clones. Re-factoring opportunities were evaluated using: (1) classification of differences involving superficial differences, signature changes, and type changes, (2) number of differences, and (3) size of candidate clones. Selected clones were automatically re-factored using ”strategy” and ”template” design patterns. Experimental evaluation was performed on JDK1.1.5 from Sun Microsystems.

Significant work followed over the next years addressing problems that include matching algorithms, scalability, and integration of clone detection in software engineering activities such as maintenance, evolution, and re-factoring. Several interesting surveys can be found in the literature together with a list of problems, many of which remain open today. In particular, recent work on software clones included new approaches to clone detection based on prefix and suffix trees, approaches to detection involving source code analysis based on latent semantic analysis, and clone identification techniques using analysis of program dependence graphs. In other works, a canonical representation of clones was developed and used for matching and comparison; interesting discussions about harmfulness of clones have also been reported; and empirical studies and evaluations of clone detection approaches can be found in several research papers. Evolution aspects have been taken into consideration in terms of evolution of clones and their lifetime over several versions of a system and in terms of software evolution by computing various similarity measures between versions. Clone research has also touched upon several interesting applications: intellectual property issues such as license infringement and plagiarism of source code have been addressed using software similarity concepts; incremental approaches to clone detection have been investigated; clones and similarity between structured software artifacts such as trees and graphs has been introduced; detection of bugs caused by inconsistent modifications between clones in a systems and between fragments in several software releases has been investigated; domain specific clones have been studied; and approaches for clone visualization have been proposed. Finally, new specialized workshops and conferences on clones and on mining software repositories have been organized.

There are many open problems that remain and possible areas for future work in CLAN (CLone Analysis) toolset including the definition of clones; addressing type III (similar) and simple type IV (semantic) clones; performance and scalability aspects; taxonomies of clones; clone classification and statistics including frequent patterns of similarity in large systems; inconsistent modifications of clones in one version of a system and inconsistent source code changes over several versions of a system leading to a taxonomy of identifiable bugs; clone matching by parallelizing and implementing it on a Graphical Processing Unit (GPU); intellectual property and plagiarism detection using spectral clone analysis; increase recall while maintaining precision; clone maximality issues under thresholds; and more.

## 2. Acknowledgements

The authors wish to thank Bell Canada for the original funding of the software quality assessment project that included clone detection and re-factoring. Recent and current research has been funded by the Natural Sciences and Engineering Research Council of Canada under the Discovery Grants Program.