Abstract:

Code clones play a major role in software maintenance and reuse. Existing code clone detection techniques mainly focus on detecting similar code fragments, methods, functions, or files. But, many design-level similarity patterns appear as recurring configurations of collaborating components such as methods, functions, classes, or any physical or logical groups of program entities. We call such types of recurring configurations as collaborative patterns. Collaborative patterns often represent program structures exhibiting specific behavior meaningful to developers who need to understand programs, reengineer legacy code for reuse, or to refactor or simply maintain programs. Among others, detection of collaborative patterns can help in change impact analysis, code compaction, and creating generics. Unfortunately, unless manually documented, collaborative patterns remain implicit in code. In this thesis, in addition to properly define the concept of collaborative patterns, we present novel methods for detecting and managing them. The main novelty of the research lies in the formulation of the concept of collaborative patterns, in the development of the technique for detecting collaborative patterns, and in the development of the technique for managing such patterns in software systems.

In the thesis, we first formalize the concept of collaborative patterns. We show possible classification of collaborative patterns. Next, we present an approach for detecting these collaborative patterns. The proposed approach for detecting collaborative patterns enhances the value of similarity analysis in the activities such as software maintenance and in the process of re-engineering for reuse that involves finding candidate modules for reuse in legacy code. Collaborative patterns are higher-level clones of large granularity that can be detected by systematically combining small-granular cloned program entities at various levels. They signify use of standardized solutions and/or repetitions that arise naturally in software analysis or design space. As such, collaborative patterns often embody important design information. Since, many existing low-level clones of small granularity are grouped around these high-level repetitions. Therefore, collaborative patterns form a convenient
conceptual window for developers to understand overall cloning in software systems. We implemented the proposed approach for collaborative pattern detection into a tool called COPAD (Collaborative Patterns Detector). We performed experimentation to evaluate the proposed approach. Finally, we propose a methodology to manage such types of high-level clones of large granularity (collaborative patterns as well as other large-granular code clones) by presenting a meta-programming technique and tool, the ART (Adaptive Reuse Technique). It manages families of redundant software systems by providing a common base of non-redundant, adaptable, and reusable meta-components called ART templates. We also evaluated the benefits of managing clones using the ART.