# Software Design Improvement through Anti-patterns Identification

Tie Feng Jiachen Zhang Hongyuan Wang Xian Wang
*College of Computer Science and Technology, Jilin University, Changchun, 130012, P.R.China*

## Abstract

*In this paper, a software design improvement approach through anti-pattern identification by case based reasoning is proposed to improve software quality and maintainability. First of all, XML based design template at micro-architecture level is presented to formally define patterns and anti-patterns. Secondly, according to 4R model of CBR, the retrieve, revision, reuse and retaining of design improving cases are illustrated. Especially, similarity measurement methods of class diagrams, sequence diagrams, OO quality metric facts and semantic constraints are proposed to identify problematic inflexible anti-patterns and replace them with high quality design. Finally, the architecture of refactoring environment developed to support this approach is introduced.*

## 1. Introduction

Identifying inflexible architecture in a given design and replacing them with high quality design alternatives has been a significant means to improve software maintainability and quality.

After some refinement and instantiation, patterns and anti-patterns have proper granularity and abstract degree to be applied to or found in software design. Applying CBR, the identification of anti-patterns and substitution with patterns can be automatically accomplished.

## 2. XML based design template

In our approach, XML is used to describe static structure and dynamic behavior of software system. Classes, relationships between classes, including inheritance, aggregation, association and delegation, and interaction sequences are defined with XML DTD. With this definition standard, patterns and anti-patterns can be easily described.

## 3. Design improvement by CBR

(1) The representation of a design improving case consists five attributes: anti-pattern description (**APD**), semantic constraints (**SC**), quality metrics before improving (**QMBI**), pattern description (**PD**) and quality metrics after improving (**QMAI**).

(2) According to the structure of case library and case representation, case retrieval consists of 4 steps: class structure graph match, object structure graph match, quality metric data match and semantic constraints match.

(3) In FACADE-oriented case reuse algorithm, an encapsulation class called FACADE is introduced, as reusing the cases, not only to hide inside micro-architecture information but also to supply an interface to invoke the component.

(4) The adjustment mainly embodies modification to weights of edges in class structure graph, object structure graph and semantic constraints.

## 4. CBR based design improvement tool

The architecture of CBDIT includes four sub-systems: UML editor, design extraction, DB and KB management and CBR reasoning engine, and three libraries: case library, semantic constraints rule library, and synonym library.

## 5. Conclusion

In order to improve the software system flexibility for adapting to future requirement change and expansion, this paper presents a Case Based Reasoning (CBR) approach to identifying micro-architecture anti-patterns and replacing them with "good" patterns.

Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM'04)
1063-6773/04 $20.00 © 2004 **IEEE**