Detection of Chained Clone and Its Application

Norihiro Yoshida
NAIST / Osaka University, Japan

Overview of my presentation

- Introduction of chained clone detection
  - Basically, it is proposed for refactoring support
- Discussion on other application of chained clone detection
  - We would like to try to apply chained clone detection into supporting other software maintenance activity.

Refactoring

- Refactoring[1] is a way to deal with code clone problem.
- Refactoring is a technique for restructuring an existing code
  - Alter software’s internal structure without changing its external behavior
  - Improve the maintainability of software
  - Number one in the stink parade is duplicate code


Difficulty of Refactoring

- It is difficult to identify refactoring opportunities in large scale source code.
  - Where are code fragments that should be merged into one method?
  - How should they be merged into one method?
    - Extract Method or Pull Up Method Refactoring?

Token-based clone detection for refactoring support (1/2)

- In many cases, Type2 clone refactoring is easier than Type3 one.
  - Type2 clone set is consist of continuous token sequences
    - It is easy to merge it into one module.
  - Type3 clone refactoring is comprised of more complicated steps
    - It needs to solve syntax differences between code fragments.

- Scalability of detection
  - Token-based clone detection tool is more scalable than syntax-based or semantic-based tools

Token-based clone detection for refactoring support (2/2)

- Basically, a set of type2 clones DO NOT have semantic similarity.
  - However, target clones for Extract Method or Pull-up Method should be semantic unit.
  - In this context, semantic clone detection is more suitable for refactoring support.
- Most token-based clone detection tools (e.g., CCFinder) DO NOT perform inter-procedural analysis.
  - One functionality is sometimes implemented by a chain of methods.
Proposed tool: Chained clone detection tool

- Detection of clone sets connected by callee-caller relations
- Scalable detection by analyzing only code fragments in CCFinder's output
  - Call-caller relations are inferred by static analysis

### Research Goal

- Define a set of clone sets having callee-caller relations as a chained clone
- Suggest applicable refactoring pattern for each chained clone based on chained clone categorization

#### Definition of chained clone (1)

- **Chained Method**
  - A set of methods that hold callee-caller relations
- **Chained Method Graph**
  - A node represents a method
  - An edge represents a callee-caller relation

#### Definition of chained clone (2)

- **Chained Clone**
  - For 2 given chained methods CM1 and CM2, we transform them into chained method graphs G1 and G2.
  - For G1 and G2, if the following three conditions are satisfied, we call the pair of CM1 and CM2 as a chained clone.
    1. G1 and G2 are isomorphic.
    2. Each pair of the corresponding nodes between G1 and G2, holds a clone relation.

- **Chained Clone Set**
  - An equivalence class of chained clones

#### Applicable Refactorings for Chained Clones

- The following refactoring[1] can be applied to merge chained clones.
  - **Pull Up Method Refactoring**
  - **Extract Method Refactoring**
  - **Extract Super Class Refactoring**

- Depending on the hierarchy relationship among Java classes having chained clones, we provide appropriate refactoring for each chained clone.
  - All chained clones in a chained clone set is in single class
  - All chained clones in a chained clone set is in multiple classes that have common parent classes
  - **Pull Up Method Refactoring** is appropriate

---

We applied suggested refactorings to chained clone sets in ANTLR.

**Case Study**

**Detected chained clone sets (Open source software)**

- ANTLR 2.7.4
- JBoss 3.2.6

**Case Study**

**Detected chained clone sets (Commercial software)**

- X
- Y

**Objective**

- How many chained clone sets exist in actual Java programs?
- Is it possible to classify chained clone sets and to apply suggested refactorings to them?

**Target software**

- Open source software
  - ANTLR 2.7.4 (47,000 LOC, 285 Classes)
  - JBoss 3.2.6 (640,000 LOC, 3364 Classes)

**Other applications of chained clone detection**

- Automated defect detection by checking the consistency of chained clones

**Why not cloned? (Defect?)**
Other applications of chained clone detection

- Precise and scalable calculation of clone ratio between methods or classes
  - Take into account whether callee methods are cloned

Previous calculation is performed from just target code

Proposed calculation takes into account callee methods

Clone set A
Method a1
Method a2

Clone set B
Method b1
Method b2

Clone set C
Method c1
Method c2

Summary

- We focus on refactoring for *chained clones* that consist of sets of the methods with callee-caller relations
  - Define chained clone
  - Method to classify chained clones according to their applicable refactorings
  - OSS and Industrial case studies

Future Works
- Apply our proposed method to other Java programs
- Other applications of chained clone detection