A Tool Support to Merge Similar Methods with a Cohesion Metric COB

Masakazu Ioka[1], Norihiro Yoshida[2], Tomoo Masai[1], Yoshiki Higo[1], Katsuro Inoue[1]
[1]Graduate School of Information Science and Technology, Osaka University, Japan
{m-ioka, t-masai, higo, inoue}@ist.osaka-u.ac.jp
[2]Graduate School of Information Science, Nara Institute of Science and Technology, Japan
yoshida@is.naist.jp

ABSTRACT
"Form Template Method" is a refactoring pattern to merge similar Java methods with syntax differences. In this refactoring, developers divide target similar methods into a template method and primitive methods corresponding to the common part and the differences, respectively. In this proposal, we present a tool to show candidates of appropriate divisions between the common part and the differences based on a cohesion metric COB when developers select a pair of similar methods.

Categories and Subject Descriptors
D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Restructuring, reverse engineering, and reengineering*

General Terms
Experimentation

Keywords
Code Clone, Refactoring, Template Method Pattern

1. INTRODUCTION
Code clone is a code fragment that has identical or similar fragments to it in the source code[3]. It is regarded as one of factors that makes software maintenance more difficult. When developers modify a code fragment, they have to find code clones corresponding to modified code fragment.

Clone refactoring (i.e., merging code clones) is a disciplined technique to reduce code clones[1]. Syntactically identical code clones can be merged by straightforward technique (e.g., Pull-Up Method refactoring, Extract Method refactoring). On the other hand, when code clones have syntactic differences, it is necessary to extract those differences as new functions (e.g., Java method, C++ function) before merging.

"Form Template Method"[1] is a common refactoring to merge a similar pair of Java methods with syntactic differences. In the refactoring, developers divide similar methods into a template method and primitive methods corresponding to the common part and the differences, respectively.

However, it is difficult for developers to identify the common part and the difference from similar methods, and extract primitive methods so that each of them has a functionality that can be given suitable method name. Therefore, tool support is needed for desirable evolution of a pair of similar Java methods with syntactic differences. Juillerat et al. proposed an approach of automatic "Form Template Method"[2]. This approach detects different subtrees by comparing sequences of AST nodes which are generated by using post-order traversal, and shows only a candidate of "Form Template Method" for each pair of similar methods regardless of satisfying developers.

In this proposal, we present a tool to show candidates of appropriate divisions between the common part and the differences based on a cohesion metric Cohesion of Blocks (COB) [5] when developers select a pair of similar methods.

2. PROPOSED TOOL
First, we explain COB. COB is a cohesion metric between block statements in source code. It is proposed by Miyake et al. for identification of a set of block statements suitable for Extract Method refactoring. Proposed tool uses COB to see whether or not expanded fragments should be extracted as primitive methods, and then suggests pairs of code fragments with high COB as excellent candidates of pairs of primitive methods. The definition of metric COB is as follow:

$$COB = \frac{1}{b}\frac{1}{v}\sum_{j=1}^{v}\mu(V_j) \quad (0 \leq COB \leq 1)$$

where:

- b is the number of code blocks,
- v is the number of used variables in the method,
- V_j is j-th variable used in the method,
- $\mu(V_j)$ is the number of code blocks using variable V_j.

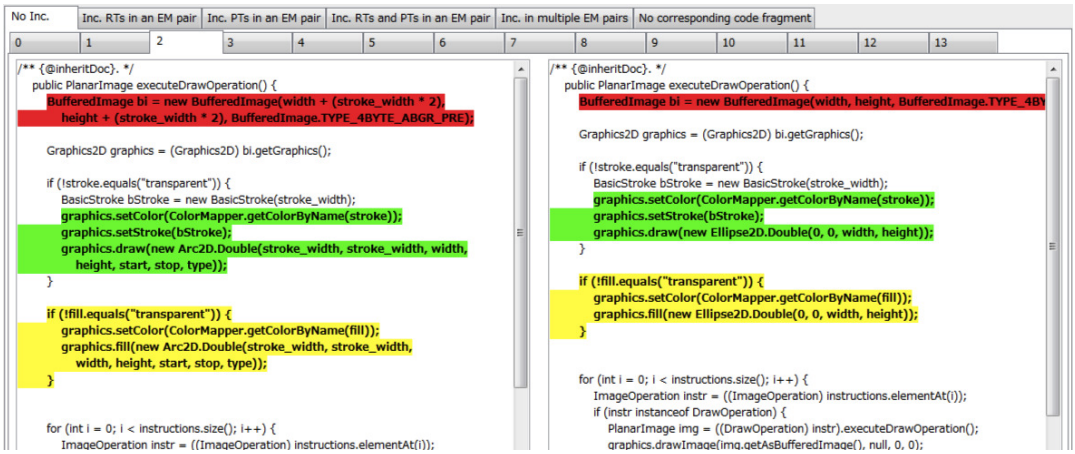

Figure 1: A screenshot of proposed tool

Next, we explain proposed tool. Proposed tool shows candidates of appropriate divisions between the common part and the differences when developers select a pair of similar methods.

Figure 1 shows a screenshot of proposed tool. Highlighted code fragments represent candidates of primitive methods. A pair of regions painted the same color means a pair of corresponding differences (i.e., candidates for primitive methods having the same name). Non-highlighted regions mean code fragments have an identical fragment to it in the corresponding method (i.e., candidate for a template method).

The steps to derive candidates of primitive are as follows.

1. Detect code fragments corresponding differences between Abstract Syntax Trees (ASTs) of given similar methods
2. Expand detected code fragments into code fragments including above, below or parent statements until all of expanded fragments are identified as extractable by the refactoring features of Eclipse JDT
3. Rank expanded fragments based on COB metric.

3. DEMONSTRATION

We applied proposed tool to the method pair in Figure 1.

This method pair is *genErrorHandler* method in CppCodeGenerator class and *genErrorHandler* method in JavaCodeGenerator class in ANTLR 2.7.4 [1]. Those methods are very similar to each other.

Figure 1 shows one of candidates for "Form Template Method" refactoring suggested by proposed tool. This candidate is highly ranked by COB metric with 0.86 because each difference shares values between blocks (COB of a method including only a block always indicates 1.0). Each highlighted code fragment has a single functionality that can be given suitable method name. Therefore, this can be considered as an excellent candidate for "Form Template Method" refactoring. Using proposed tool with COB based ranking, developers are possible to find appropriate candidates of "Form Template Method" refactoring easily.

4. SUMMARY AND FUTURE WORK

We proposed a tool to show candidates of template primitive methods for "Form Template Method" refactoring, and demonstrate it. As future work, we are planning to use cohesion metrics based on program slicing [4] instead of metric COB, because we expect ranking is better using program slicing. Also, we will implement the code transformation for "Form Template Method" refactoring using the Language Toolkit of Eclipse Project [2].

Acknowledgments

This work is partially supported by JSPS, Grant-in-Aid for Scientific Research (A) (21240002) and Grant-in-Aid for Research Activity start-up(22800040).

5. REFERENCES

[1] M. Fowler. *Refactoring: Improving the Design of Existing Code*. Addison Wesley, 1999.

[2] N. Juillerat and B. Hirsbrunner. Toward an Implementation of the "Form Template Methodï£¡h Refactoring. In *Proc. of SCAM 2007*, pages 81–90, Paris, France, 2007.

[3] T. Kamiya, S. Kusumoto, and K. Inoue. CCFinder: A multilinguistic token-based code clone detection system for large scale source code. *IEEE Trans. Softw. Eng.*, 28(7):654–670, 2002.

[4] T. M. Meyers and D. Binkley. An empirical study of slice-based cohesion and coupling metrics. *ACM Trans. Softw. Eng. Methodol.*, 17:2:1–2:27, December 2007.

[5] T. Miyake, Y. Higo, and K. Inoue. A software metric for identifying extract method candidates. *IEICE Trans. Inf.& Syst.(Japanese Edition)*, J92-D(7):1071–1073, 2009.

[1] http://antlr.org/

[2] http://www.eclipse.org/articles/Article-LTK/ltk.html