Extracting collaboration classes from Java source code

Nguyen Van Tuan (0810021)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 9, 2010

Keywords: Collaboration Classes(CC), Design Pattern, Metapattern, UML, Java, Class Diagram.

1 Background and Purpose

When developing an information system, there is an enormous amount of software models and programs. And the dependency relationships between them are very complicated. Therefore, tools to help automate the change process are very useful in, reducing the effort required to change and improving reliability of information system. Moreover, automatically detecting collaboration classes that correspond to the elements of the design model plays an important part in this tool.

The purpose of this research is to identify collaboration classes which are Java classes implementing a use case and extract them.

2 Related work

In 2006, Kim had defined collaboration classes is Java classes that use a metapattern and classified 6 types of metapattern of Pree to 3 types of coordinative structure and had developed an algorithm to extract them from Java source code. His purpose is to develop an algorithm that can extract 23 types of GoF design patterns, but he could not extract all of them. His algorithm can extract only 17 types of them. In 6 types of

design pattern that his algorithm cannot extract, 3 types can be described by metapattern and the others are not.

3 Research approach

To extract collaboration classes, we improve on previous research, and use it to extract Java classes that implement a use case. In order to solve this problem, there are two issues.

1. Adding some methods to extract Java classes which use design pattern and cannot be explained by the metapattern. Specifically, besides using metapattern we use structural features and behavior features of design patterns to develop an algorithm to extract Java classes using design pattern.

2. Extracting Java classes that corresponding to classes in class diagram of use case.

For issue 1, when implementing a class diagram, developers usually use design pattern to implement. Therefore, extracting classes that applied design patterns plays an important part in extracting collaboration classes. And because of metapatterns that defined by Pree can explain almost of design patterns, we investigate design patterns that Extract algorithm developed by Kim cannot extract and improve it to be more effective (can extract all cases which can be explained by metapattern). Moreover, by summarizing structural features and behavior features of design patterns that cannot be explained by metapattern we develope an algorithm to extract classes that are applied design pattern and cannot be explained by metapattern.

For issue 2, by applying algorithm talked in issue 1 we can extract classes that applied design pattern from Java source code. And using constructional feature of design pattern in extracted classes, we extract Java classes that correspond to individually classes in class diagrams of use cases. After that, we apply subgraph isomorphism algorithm to find correspondences between classes in class diagrams and extracted Java classes. However in some cases, because developers do not use design pattern to develop system,

using only constructional features of design pattern in extracted classes is not enough to get exact results. Therefore, we propose a rule for tracking source codes that are not be applied design pattern when implemented.

4 Experiment

To check effective of extracted classes that were applied design pattern from Java source code, we checked it with 41 design patterns that appear on Pattern in Java book of Mark Grand and used source codes that were included this book 's CD. As a result of test, this algorithm can extract classes that applied 39 design patterns in the book.

Besides, we checked the effective of our approach by using two minor systems. It is ATM system and elevator controller system. And it exactly extracted over 92.5% classes that included in use cases.

5 Conclusions and Future work

This research improved algorithm proposed by Kim and added some structural features and behavior features of design pattern's source code to extract more design patterns than Kim proposed. Based on exist algorithm, we develop an algorithm to extract classes that included in use cases by using constructional features of extracted class and applying subgraph isomorphism algorithm proposed by J R Ullman.

In the future, we will consider the failing; apply method to other design patterns and other programming languages.