Liu, C.-C. et al. (Eds.) (2014). Proceedings of the 22nd International Conference on Computers in Education. Japan: Asia-Pacific Society for Computers in Education

Learning Method for Understanding Design Policy of Object-oriented Design and its Meta-learning Support System

Tomoko KOJIRI[a*], Hiroki OOE[b], & Kazuhisa SETA[c]
[a]*Graduate School of Science and Engineering, Kansai University, Japan*
[b]*Faculty of System Science and Engineering, Kansai University, Japan*
[c]*Graduate School of Science, Osaka Prefecture University, Japan*
*kojiri@kansai-u.ac.jp

Abstract: Design pattern is a description of the object-oriented design that gives high-quality and re-usable solutions for frequently occurring problems. A design pattern is formed based on the various experiences of predecessors who made ineffective designs. If students can grasp the design policy of predecessors from the design pattern, they can create good designs and apply them to new problems. The objective of this research is to propose a learning method for understanding design policy of the design pattern and to construct a meta-learning system for proposed learning method.
Keywords: design policy, experiential knowledge, meta-learning support system

1. Introduction

Design patterns are experiential knowledge for designing better object-oriented programs. They have re-usability and extensibility advantages, so a policy of constructing re-usable and extensible designs in object-oriented designs can be found in the process of forming design patterns. If students can follow the processes of the predecessors who created design patterns, they might learn such good design policy. However, most textbooks of design patterns only describe their definitions and show how to use them (Gamma et al. 1994, & Freeman et al. 2004); since the creation process is not clearly mentioned, students are not able to consider the design policy.

To understand design policy, we believe that following the thinking process of predecessors is effective. The objective of this research is to support students to understand design policy by introducing a new learning method that repeatedly creates ineffective design from the good design and compares effective and ineffective designs under the various conditions. This activity corresponds to the part of the predecessors' processes of creating design patterns to give students a chance to identify good design policy. Some traditional researches give problems that are solved by design patterns and encourage students to create design patterns by themselves (Weiss 2010 & Pillay 2010). This approach is effective if students can derive design patterns by themselves. However, many have difficulty creating them from scratch. Therefore, in our approach, a program based on a design pattern (*design pattern program*) is given and students transform the given program into an inappropriate one (*alternative solution*). Then they compare the design pattern programs and the alternative solutions for a new problem (*extended problem*), which is an extended problem of the original one. By repeating this process and considering the alternative solutions with several different extended problems, the design pattern's design policy can be experientially acquired.

2. Approach

When encountering a problem, predecessors created various solutions by trial-and-error and evaluated the quality of their solutions based on how much they need to be modified based on extended problems. During the trial-and-error process, good solutions are selected based on the predecessors' criteria and the final solution becomes a design pattern. Special structures in each design pattern, such as the types of classes and their relations, reflect design policy. Therefore, to replace the special structure in the design pattern program with another structure that can perform the same behavior may lead students to consider the following questions and to learn the design policy of predecessors.

- What special structure is embedded in the design pattern program?
- What kinds of problems can be solved by the design pattern?

Figure 1 illustrates our learning method for understanding design policy. From the given design pattern program, students create alternative solutions. The inappropriateness of the alternative solutions and the effects of the design pattern program are evaluated under the extended problems.

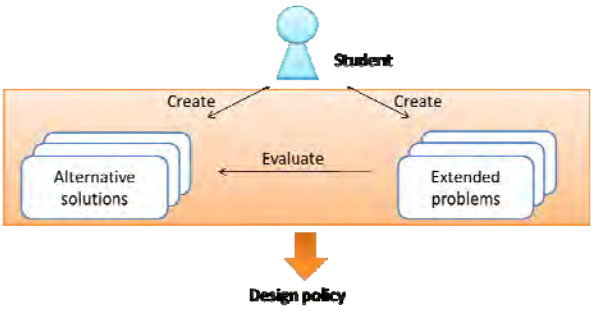

Fig. 1 Method for learning design policy

Since students are generally not familiar with the learning method proposed in Fig. 1, creating alternative solutions may be difficult. This research proposes a meta-learning support system of the learning method proposed in Fig. 1. In the meta-learning support system, the learning process is divided into two phases, such as alternative solution creation phase and solution evaluation phase, so that students can focus on individual phases. The alternative solution creation sub-system, which supports the creation of alternative solutions from the given design pattern program, holds several alternative solutions as expected answers. It examines the alternative solutions created by students and leads them to derive expected answers. On the other hand, the solution evaluation sub-system gives an example of extended problems so that students can evaluate the qualities of the design pattern program and the alternative solutions. This sub-system is invoked immediately after students have created alternative solutions in the alternative solution creation sub-system. Extended problems are given as multiple-choice questions (Table 1) that ask the part of the programs that should be modified under the extended problems. These questions themselves promote the students' activities of considering the design policy.

Table 1 Example of extended problem for *Adapter pattern*

[Extension of problem] Change the *Adaptee* class with *methodA* to *Adaptee2* class with *methodB*. Consider which parts you need to modify in both class diagrams.

[Answer] You only need to [1] [2] in class [3] for a design pattern program, but you have to [4] [5] in class [6] for an alternative solution.

[Choices for each blank]

1, 4: Type of modification: change, add, or eliminate

2, 5: Target of change, e. g., names of the method, the instance, or the constructor

3, 6: Name of existing class

3. Alternative Solution Creation Sub-system

The alternative solution creation sub-system provides a supporting environment for creating alternative solutions from the given design pattern program. Since its structure in the object-oriented design can be illustrated by a class diagram, our system gives the design pattern program as a class diagram and provides an interface in which the given class diagram can be changed (Fig. 2). The given class diagram is shown on the class diagram display unit. The class diagram can be edited by inputting the class name or the relation name or selecting the class or the relation to remove from the class diagram edit unit. When the re-illustration button is pushed, the class diagram that satisfies the inputted classes and relations is depicted in the class diagram display unit.

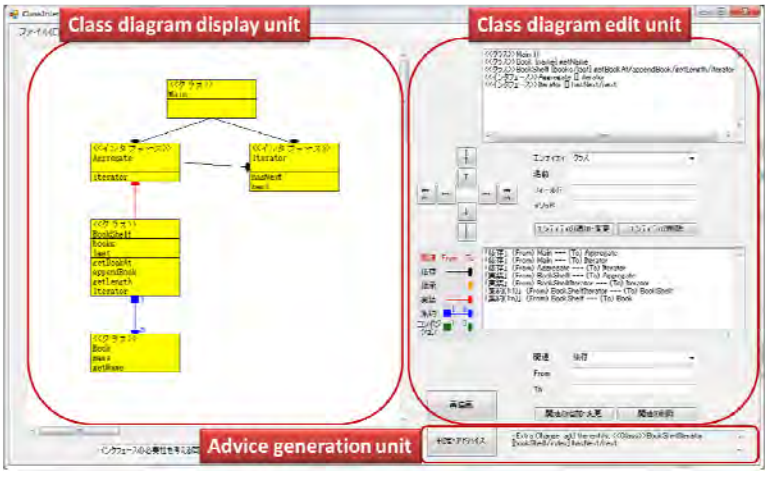

Fig. 2 Interface of alternative solution creation sub-system

The system holds the class diagrams of the expected alternative solutions and evaluates those created by students. If a student's class diagram does not match its alternative solutions, advice is generated in the advice generation unit that indicates the differences of the entities or the relations from the expected alternative solution.

4. Solution Evaluation Sub-system

After a student successfully creates one of the expected alternative solutions, the solution evaluation sub-system starts. It displays two windows. One show the extended problem, which consists of problem and answer sentences with blanks and lists of answer choices (Fig. 2). Students need to select one choice for each blank from the list. The other window illustrates the class diagrams of the design pattern program and alternative solutions created by students (Fig. 3). This window is used for comparing the quality of the design pattern program and the alternative solution under the given extended problem. Students can invoke edit windows for freely changing these class diagrams. In addition, by clicking on the classes in the class diagrams, the programs of the classes are displayed.

The system holds the extended problem answers. When students push the answer button, it evaluates the answer and gives advice, if necessary. Currently, three kinds of advice are prepared. Since it is effective to create class diagrams that are modified based on the extended problem, the first piece of advice explains how to modify the class diagrams. The advice is shown in the advice generation unit in Fig. 3. If students are still not able to derive the answer, the system indicates the part of the program that should be changed under the extended problem as the next piece of advice. If students are not able to answer correctly based on the second bit of advice, the system provides text that points out the incorrect choices and explains how to select the right choice

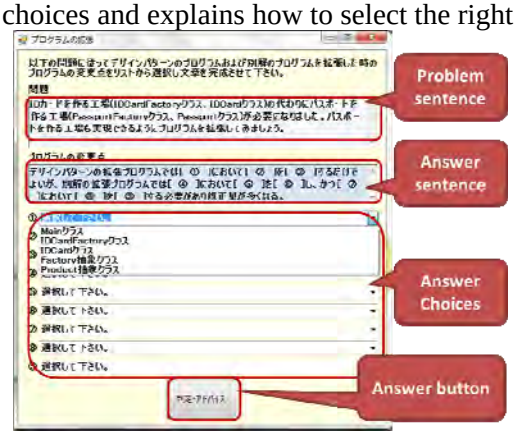

Fig. 2 Window displaying extended problem

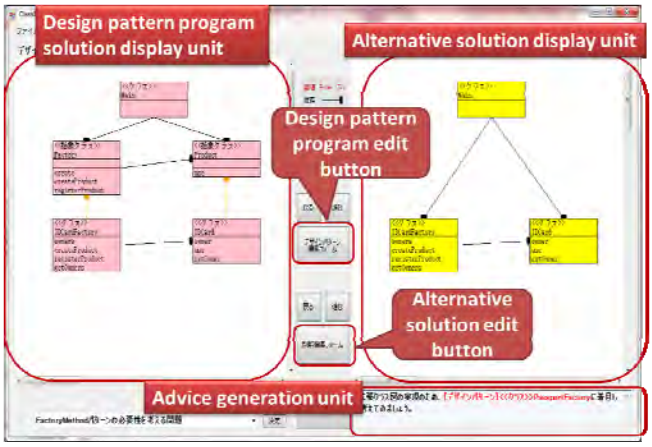

Fig. 3 Window showing class diagrams of design pattern program and alternative solution

5. Conclusion

In this research, we proposed a learning method to acquire design policy by redesigning a design pattern program to reach alternative solutions. In addition, we constructed a meta-learning system in which students can learn the proposed learning method. The alternative solution creation sub-system supports the creation of alternative solutions from the design pattern program. On the other hand, the solution evaluation sub-system supports the evaluation of alternative solutions and the design pattern program by providing extended problems.

In the current system, extended problems are given by the solution evaluation sub-system. However, knowing how to extend the original problem is also important for evaluating designs. The ability to create appropriate extended problems is also worth supporting. Therefore, in future work, we need to extend our system to support trial-and-error activities not only to create alternative solutions but also to derive appropriate extended problems.

Acknowledgements

This work was partly supported by the Telecommunications Advancement Foundation.

References

Freeman, E., Robson, E., Bates, B., & Sierra, K. (2004). Head First Design Patterns, *O' Reilly Media*.

Gamma, E., Helm, R., Johnson, R., & Vissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software, *Addison-Wesley Professional.*

Pillay, N. (2010) Teaching Design Patterns, *Proc. of SACLA.*

Weiss, S. (2010). Teaching Design Patterns by Stealth, *Proc. of SIGCSE 2005*, 492-494.