

UnderFrame: Understanding Object-Oriented Frameworks Using a Case-Based Teaching Approach *

Guillermo Jiménez-Díaz, Mercedes Gómez-Albarrán, Pedro González-Calero

Departamento de Sistemas Informáticos y Programación
Universidad Complutense de Madrid
Juan del Rosal 8. 28040 Madrid, Spain
gjimenez@fdi.ucm.es, {albarran, pedro}@sip.ucm.es

Abstract. Active learning and example-based methods are the main ideas behind effective techniques for object-oriented framework understanding and learning. This paper describes our ongoing work on a Case-Based Teaching approach for learning frameworks that follows these ideas.

The approach is based on the involvement of the student in the resolution of exercises selected according to her knowledge level. It has the following characteristics: it provides information to help in the exercise resolution and to understand the solutions; it monitors the student actions, detecting wrong actions; and it lets analyze and test student's solutions through scenario diagram simulations.

The paper briefly describes the interactive visual learning environment, the exercise content and the kind of information provided to the student.

*Supported by the Spanish Committee of Science & Technology (TIC2002-01961).

Learning to effectively use an object-oriented framework takes a long time. Till moment, little work has been done on suggesting effective techniques that would help to reduce this time. Some researchers have focused on developing effective documentation resources that could improve the self-education process [1]. Other works suggest a training process where a human tutor guides the learner following example-based techniques and an incremental learning process supported by framework architecture knowledge [2]. Apart from this, effective techniques to teach frameworks should take into account that “nobody understands a framework until they have used it” [3].

This paper presents our ongoing work on a Case-Based Teaching approach to teach frameworks. Case-Based Teaching (CBT) [4] is a Learning by Doing approach where the student works on a task that pursues a pedagogical goal. When the student fails, she is best suited to learn and examples can be provided to resolve her misconceptions. So CBT gathers two basic principles for learning frameworks: example-based techniques and active learning methods. We are developing a CBT methodology to teach frameworks where the student learns in an incremental way resolving more and more complicated exercises to practice the main concepts from the framework [5]. This CBT methodology extends our previous work on case-based framework documentation that facilitates the framework self-education process [6]. The pedagogical ideas of the methodology are applied to develop UnderFrame (UNDERstanding FRAMEworks), a CBT system for teaching frameworks, whose first prototype is dedicated to JHotDraw [7].

Visual Environment

UnderFrame involves students in the development of a sample application using the framework. To avoid the complexities of dealing with source code, we have defined a set of instantiation actions, implementation pieces that can be used to extend any framework (Table 1). In UnderFrame, students should use the instantiation actions in order to implement the application. Each instantiation action is translated into source code with a template.

Our experience using frameworks has shown that the development with a framework consists of a “trial and error” process: we run the developed application and we make a trace of it when we find mistakes, until we consider we have an application that satisfies the requirements. UnderFrame provides the student a simple and attractive way to test her application using scenario diagram simulations. A scenario diagram is a notation for visualizing the message flow in object-oriented systems. We are developing anthropomorphic dramatizations from these diagrams, where each class is represented by an actor who plays the role of this class in framework interactions. Besides, the student can take part in the simulation, playing the role of a class. The simulation lets the students practice skills repeatedly until they are well-honed, understanding the roles of the classes within the framework. We have extended the diagrams to enhance the information that they can provide to the student during the simulation. On the one hand, the simulation can provide hints about which instantiation actions are wrong to the resolution of the exercise and how these actions affect the application. On the other hand, scenario diagrams are linked to additional

information about framework design, so the student could understand the behavior and the interactions among the classes within the framework.

Table 1. Instantiation Actions

Instantiation Action	Description
<i>CreateClass</i>	It creates a new class extending another one.
<i>ImplementInterface</i>	It creates a new class implementing an interface.
<i>CreateConstructor</i>	It creates a new constructor for a class.
<i>CreateMain</i>	It creates a main method in a class.
<i>CreateMethod</i>	It creates a new method in a class.
<i>UpdateMethod</i>	It rewrites or specializes a method in a class.
<i>CreateObject</i>	It creates a new object.
<i>CallMethod</i>	It calls an object method.
<i>CallSuper</i>	It calls a method of the superclass.
<i>AddCode</i>	It adds a code fragment in a method.

Exercises

UnderFrame contains an exercise base indexed by the framework concepts that students should learn. These concepts depend on the framework domain. For instance, we have created for JHotDraw an index according to the concepts described in the JHotDraw Pattern Language developed by Douglas Kirk [8]. This index is used by the retrieval methods in order to advise the student the next exercise she should resolve according to her framework knowledge.

An exercise in UnderFrame contains several kinds of information:

- A description about what the student should develop and a list of the framework concepts that the exercise meets.
- The exercise solution in terms of instantiation actions. We are working on a structure similar to Petri nets for depicting the execution order of the instantiation actions, including information about the sets of actions that can be executed in an unordered way.
- Help information about the architecture and behavior of the framework. On the one hand, each exercise contains framework micro-architectures [9], small pieces from the complex global architecture that make easier the understanding of the framework design to the student. We have added to these micro-architectures another relevant documentation, such as JavaDocs, so the student understands the actions that should use to resolve the exercise. On the other hand, the exercise contains the scenario diagrams that will be useful for “trying” it during the “trial and error” process. It is also necessary to include information about how each instantiation action affect when passing messages during the scenario simulation.

Providing Help Information

CBT and Learning by Doing are approaches based on letting the student make mistakes because at this moment the student is best suited to learn. UnderFrame follows this approach, although it can also provide information when the student asks for it. UnderFrame provides information when:

- The student resolves an exercise. At the highest level, UnderFrame can provide information design contained in the micro-architectures, JavaDocs and in the classes that participate in the simulation. If needed, UnderFrame can provide prescriptive information about the implementation, that is, steps about the classes and methods the student should create or modify. UnderFrame can also resolve the exercise automatically step-by-step using the solution contained in the exercise.
- The student tests the exercise with the scenario simulation. If the student solution is wrong, the corresponding scenario simulation stops and UnderFrame provides hints about which instantiation actions have caused the mistake.

The CBT technology is part of a doctoral dissertation research. Nowadays we are working on the development of a set of useful exercises to cover the main functionality of the framework JHotDraw and the design of the interface of the visual environment. We have also produced a small prototype about the scenario diagram simulation.

References

1. Butler, G., Dénomée, P.: Documenting Frameworks. In: Fayad, M., Schmidt, D., Johnson, R. (eds.): Building Application Frameworks, John Wiley and Sons, New York, (1999) 495-503.
2. Eckstein, J.: Empowering Framework Users. In: Fayad, M.E., Schmidt, D.C., Johnson, R.E. (eds.): Building Application Frameworks: Object-Oriented Foundations of Framework Design, John Wiley & Sons, New York, (1999) 505-521.
3. Johnson, R.: Documenting Frameworks using Patterns. Procs. Conference on Object-Oriented Programming Systems, Languages, and Applications (1992) 63-76.
4. Kolodner, J.L.: Educational Implications of Analogy. A View from Case-Based Reasoning. *American Psychologist*, 52, (1997) 57-66.
5. Jiménez-Díaz, G., Gómez-Albarrán, M.: A Case-Based Approach for Teaching Frameworks. Accepted in. PhD-Workshop at 18th European Conference on Object-Oriented Programming (2004).
6. Gómez-Albarrán, M., González-Calero, P.A.: Applying Case-Based Reasoning to Support Dynamic Framework Documentation. *International Journal of Software Engineering and Knowledge Engineering*, 11, (2001) 479-502.
7. JHotDraw Website: <http://www.jhotdraw.org/>.
8. JHotDraw Pattern Language: <http://softarch.cis.strath.ac.uk/PLJHD/Patterns/JHDPatternIndex.html>.
9. Lajoie, R., Keller, R.K.: Design and Reuse in Object Oriented Frameworks: Patterns, Contracts, and Motifs in Concert. Procs. Colloquium on Object Orientation in Databases and Software Engineering (1994) 295-312.