On Extracting the Semantics of the Iterator Pattern For Use in a Case Tool

1Joan Peckham and 2Scott J. Lloyd
University of Rhode Island, 1Tel: (401) 874-4174, 2Fax: (401) 874-4617
1Tel: (401) 874-7056, 2Fax: (401) 874-4312, joan@cs.uri.edu, sjlloyd@uri.edu

ABSTRACT
Software patterns are used to facilitate the reuse of object-oriented designs. While most CASE (Computer Aided Software Engineering) tools support the use of UML (Unified Modeling Language) [AJO98] to extract the design from the software engineer and assist in development, most do not provide assistance in the integration and code generation of software patterns. In this paper, we analyze the Iterator software pattern [Gamma95] for the semantics that would be used in a CASE design tool to help the software engineer to integrate this pattern into the design and then generate some of the code needed to implement the pattern. This work is based upon semantic data modeling techniques that were previously proposed for the design of active databases [BMPV97, PMD95].

INTRODUCTION
One of the intents of the object-oriented (OO) programming paradigm is to assist in the reuse of code through the use of classes that bundle data structures and procedures in such a way that they could more easily be moved from one implementation to another. When OO languages were first introduced, code libraries were developed to permit the sharing of objects and classes. At the same time, OOA&D (Object-Oriented Analysis and Design) techniques were being developed [Booch94, CY91, Jacobs692, Rumbaugh91, W-BWW90]. This gave us a set of notations for expressing the design of OO applications. The library became a vehicle for the reuse of the OO designs and led to the capture of software design of designs that are frequently reused in software applications, but are somewhat independent of particular application types. These patterns are archived in books, for example [Gamma95]. A combination of text, UML, and code samples is used to communicate the patterns. Early industrial experience indicates that patterns speed the development of systems but are hard to write [Beck96] so a few CASE tools provide computer assistance to the programmer in choosing and integrating automatically generated code for the patterns in their applications.

CASE tools that support the use of software patterns include [BFVY96, FMW97, Paulisch96]. While these tools are just beginning to emerge, none have integrated code generation and general design in a generic way that permits seamless code specification with patterns. For example, the techniques are not generally language independent and are unable to generate code in more than one language. Some existing tools generate code, but into a different workspace from the general software specification and coding environment, requiring the cutting and pasting of code from the pattern code space.

All software patterns have alternative implementations. These are typically explained using text and sample code. Software engineers are then expected to use this information to construct their own implementation of the pattern. Our goal here is to capture the semantics of patterns well enough so that they can be presented to the software engineer via a named choices in the CASE tool and then be used to generate the code. Earlier in [PM00] we began to elaborate the choices in the Observer pattern of [Gamma95]. In this paper we look at the Iterator pattern from the same source.

A PATTERNS PRIMER
- The Iterator Pattern

As laid out in [Gamma95], patterns are described using diagrams, textual descriptions, and sometimes with examples using a pseudo language or code. Included in each pattern description are:
- Name - The name of the pattern
- Classification - The type of pattern based upon the authors’ classification system (there are three, creational, structural, and behavioral)
- Intent - The purpose of the pattern, what it does

This paper appears in Issues and Trends of Information Technology Management in Contemporary Organizations, the proceedings of the Information Resources Management Association International Conference. Copyright © 2002, Idea Group Inc.
CONCLUSIONS

Software patterns can be very useful to prevent software designers and developers from “reinventing the wheel” for every new utilization of a specific pattern. This can lower the cost of software development and maintenance, providing more robust and error free code. Currently software patterns have not been integrated into CASE tools in a robust manner. In this paper we have shown how we envision the integration of patterns into CASE design tools assist in the selection of the proper patterns and to partially generate the code to implement the Iterator pattern. There are numerous other applications possible for this type of software tool that will be explored in future research. For example, the pattern books outline how specific patterns can be used with each other in a system design. The CASE tool can use this information to offer assisting and cooperating patterns to the designer and begin to generate the code. It can also provide checking assistance in identifying problems or errors that are frequently made when specific patterns or combinations of patterns are selected.

ENDNOTE

1 Thank you to Heng Chen, Chen Gu, and Mingsong Zheng of Professor Peckham’s CSC 509 (Software Engineering) class for beginning to outline the semantics of this pattern in a classroom exercise for us.

Figure 1: Iterator pattern, basic structure

REFERENCES


[Gamma95] Gamma, Helm, Johnson, Vlissides, and Booch. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1995.


Related Content

Hybrid TRS-PSO Clustering Approach for Web2.0 Social Tagging System
[www.irma-international.org/article/hybrid-trs-pso-clustering-approach-for-web20-social-tagging-system/122777](www.irma-international.org/article/hybrid-trs-pso-clustering-approach-for-web20-social-tagging-system/122777)

Detection of Shotgun Surgery and Message Chain Code Smells using Machine Learning Techniques

Automatic Item Generation
[www.irma-international.org/chapter/automatic-item-generation/183950](www.irma-international.org/chapter/automatic-item-generation/183950)

Digital Literacy
[www.irma-international.org/chapter/digital-literacy/183935](www.irma-international.org/chapter/digital-literacy/183935)

Particle Swarm Optimization from Theory to Applications
[www.irma-international.org/article/particle-swarm-optimization-from-theory-to-applications/197378](www.irma-international.org/article/particle-swarm-optimization-from-theory-to-applications/197378)