IRM PRESS
701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.irm-press.com
ITB9314

Chapter I

Integrating Patterns into CASE Tools

Joan Peckham
University of Rhode Island, USA

Scott J. Lloyd
University of Rhode Island, USA

ABSTRACT

Software patterns are used to facilitate the reuse of object-oriented designs. While most Computer Aided Software Engineering (CASE) tools support the use of Unified Modeling Language (UML) (Alhir & Oram, 1998) 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 chapter, we analyze the Iterator software pattern (Gamma et al., 1995) for the semantics that would be used in a CASE-design tool to help the software engineer to integrate this pattern into a design and then generate some of the code needed to implement the pattern. This work is based on semantic data modeling techniques that were previously proposed for the design of active databases (Brawner, MacKellar, Peckham & Vorbach, 1997; Peckham, MacKellar & Doherty, 1995).

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, Object-Oriented Analysis and Design (OOAD) techniques were being developed (Booch, 1994; Coad & Yourdon, 1991; Jacobson, 1992; Rumbaugh et al., 1991; Wirfs-Brock, Wilkerson & Weiner, 1990). This gave us a set of notations for expressing the design of OO applications. The libraries also became a vehicle for the reuse of the OO designs and led to the capture of software patterns or designs that are frequently reused in software applications but are somewhat independent of particular application types. One of the most often cited book archives is *Design Patterns: Elements of Reusable Object-Oriented Software* (Gamma, Helm, Johnson, Vlissides & Booch, 1995). 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 (Beck et al., 1996). So a few CASE tools provide computer assistance to the programmer in choosing and integrating automatically generated code for the patterns in their applications. Tools that support the use of software patterns include those by Budinsky, Finnie, Vlissides and Yu (1996), Florijn, Meijers and van Winsen (1997) and Paulisch (1996). 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 that they can be presented to the software engineer via named choices in the CASE tool and then be used to generate the code. In Peckham and MacKellar (2002) we began to elaborate the choices in the Observer pattern of Gamma et al. (1995). In this paper we look at the Iterator pattern from the same source.

CASE TOOLS

CASE tools are used to assist software engineers in all aspects of the software lifecycle. These tools can help a team to manage, design, implement, test and

Automatic Timed Automata Extraction from Ladder Programs for Model-Based Analysis of Control Systems

Kézia Oliveira, Kyller Gorgônio, Angelo Perkusich, Antônio Limaand Leandro Dias da Silva (2011). *Software Engineering for Secure Systems: Industrial and Research Perspectives (pp. 305-328).*

www.irma-international.org/chapter/automatic-timed-automata-extraction-ladder/48415