Chapter 2.7
Formal Modeling and Specification of Design Patterns Using RTPA

Yingxu Wang
University of Calgary, Canada

Jian Huang
University of Calgary, Canada

ABSTRACT

Software patterns are recognized as an ideal documentation of expert knowledge in software design and development. However, its formal model and semantics have not been generalized and matured. The traditional UML specifications and related formalization efforts cannot capture the essence of generic patterns precisely, understandably, and essentially. A generic mathematical model of patterns is presented in this article using real-time process algebra (RTPA). The formal model of patterns are more readable and highly generic, which can be used as the meta model to denote any design patterns deductively, and can be translated into code in programming languages by supporting tools. This work reveals that a pattern is a highly complicated and dynamic structure for software design encapsulation, because of its complex and flexible internal associations between multiple abstract classes and instantiations. The generic model of patterns is not only applicable to existing patterns' description and comprehension, but also useful for future patterns' identification and formalization.

INTRODUCTION

Design patterns are a powerful tool for capturing software design notions and best practices, which provide common solutions to core problems in software development. Design patterns are a promising technique that extends reusability of software from code to design notions. A representative work of design patterns is initiated by

Gamma and his colleagues in *Design Patterns: Elements of Reusable Object-Oriented Software* in 1994 (Gamma, Helm, Johnson, & Vlissides, 1995). Design patterns may speed up the development process by providing tested and proven development paradigms. Reusing design patterns helps to prevent subtle issues in large-scale software development and improves code readability for architects and programmers. Design patterns can contribute to the definition, design, and documentation of class libraries and frameworks, offering elegant and reusable solutions to design problems, and consequently increasing productivity and development quality (Gamma et al., 1995; Wang, 2007a). Each design pattern lets some aspects of the system structure vary independently of other aspects, thereby making the system more robust to a particular kind of change.

Design patterns are used to be modeled and specified in natural language narratives, object-oriented programming languages, and UML diagrams. The traditional means are either inherently ambiguous or inadequate (Lano, Goldsack, & Bicarregui, 1996; Vu & Wang, 2004; Wang & Huang, 2005). The major problems in current methodologies for pattern specification are identified as follows:

- **The lack of a unified and generic architecture of patterns as a multilayered complex entity with a set of abstract and concrete classes and their interrelations:** Patterns have been classified in three categories known as the *creational*, *structural*, and *behavioral* patterns (Gamma et al., 1995). However, the theories for the nature of patterns and their generic architecture are yet to be sought.
- **The lack of abstraction:** Almost all patterns are described as a specific and concrete case in natural language, UML diagrams, or some formal notations. However, no generic mathematical model of patterns is rigorously established, which may form a deductive basis for deriving concrete and application-specific patterns.
- **The lack of uniqueness:** In the conventional pattern framework, there are different patterns that may be implemented by similar code; Reversely, the same pattern may be implemented in various ways.
- **The use of unstructured semantic means to denote highly complicated design knowledge in patterns:** The informal descriptions of patterns puzzle users and cause substantial confusions. Even the creators of patterns demonstrate inconsistent over the semantics of certain patterns.

The authors perceive that the above fundamental problems can be alleviated by introducing formal semantics for design patterns and their generic mathematical models (Wang, 2002, 2003, 2006a-c, 2007a-c). This approach allows for unambiguous specifications, enables reasoning about the relationships between abstract and concrete patterns, and promotes a coherent framework for the rapidly growing body of software patterns in software engineering (Beck, Coplien, Crocker, & Dominick, 1996; Bosch, 1996; Wang, 2002, 2006a, 2007a). This article presents a generic model of design patterns and a formal specification method for design patterns using Real-Time Process Algebra (RTPA) (Wang, 2002, 2003, 2007a). The approach proposed in this article aimed at the following objectives:

- **It is generic:** The same pattern model can be adopted to specify any existing and future pattern, particularly user defined patterns. To some extent, the general pattern model is the pattern of patterns.
- **It is formalized:** The mathematical semantics and formal notation system are based on RTPA (Wang, 2002, 2003, 2007a).
- **It is expressive:** Only 34 notations are used to denote class association relationship and specify patterns from three facets known