Chapter VIII

Formalizing and Analyzing UML Use Case View Using Hierarchical Predicate Transition Nets

Xudong He, Florida International University, USA

Abstract

Unified Modeling Language (UML), developed by a group of leading experts in object-oriented methodologies, has become the standard object-oriented development methodology in the software industry. UML contains a set of diagrams for describing different views and aspects of systems. UML use case diagrams are used during requirements analysis to define a use case view that constitutes a system’s functional model. Each use case describes a system’s functionality from a user’s perspective. However, the use case descriptions are often informal, which are error-prone and cannot be formally analyzed to detect problems in user requirements or errors introduced in a system functional model. A well-defined use case view is not only necessary for subsequent correct system design and implementation but also serves as a basis for future system evolution. Therefore, it is extremely important to ensure the correctness of the functional model captured in a use case view. In this chapter, we present an approach to formally translate a use case view into a formal model in hierarchical predicate transition nets that support formal analysis and thus are capable to detect possible requirements and modeling errors in a use case view.
Introduction

Software evolution is greatly impacted by a software design paradigm and software development methodologies. The object-oriented (OO) software development paradigm has been widely adopted in software industry in recent years and thus dictates the future software development and evolution. Among the various existing OO methods, UML (Booch, Rumbaugh, & Jacobson, 1999; Rumbaugh, Jacobson, & Booch, 1999) has become the de facto standard OO design language. UML contains a set of graphical notations for different views and aspects of software systems. While UML graphical notations are conceptually sound, they lack precise semantics and thus do not support formal analysis. Although it is desirable to separate and individually define the different aspects of a system, it is often not clear how to do and is very difficult to relate the different views together and to ensure their consistency.

To tackle the impreciseness of UML, there have been considerable research activities to make the UML semantics more precise in recent years. There is a worldwide pUML (precise UML) research group. Researchers have attempted to define formal semantics for class diagrams (Shroff & France, 1997; Evans, 1998; France, Evans, Lano, & Rumpe, 1998; Lano & Bicarregui, 1998; McUmber & Cheng, 1999; He, 2000a; McUmber & Cheng, 2001), use case diagrams (Overgaard & Palmkvist, 1998, Back, Petre, & Paltor, 1999; He 2000b), interaction diagrams Knapp 1999, and statechart diagrams (McUmber & Cheng, 1999; Saldhana & Shatz, 2000; Saldhana, Shatz & Hu et al., 2001; Dong & He, 2001; McUmber & Cheng, 2001; Dong, Fu, & He, 2003). A variety of formal methods have been applied in the previous attempts, including variants of Z, variants of logic and temporal logic, refinement calculus, and variants of Petri nets.

In this chapter, we focus on one of the UML notations, use case diagrams. Use case diagrams were proposed by Jacobson, Christerson, Jonsson, and Overgaard (1992) to capture typical system use scenarios in system analysis. UML use case diagrams are used to define a use case view that constitutes a system’s functional model. Each use case documents a system’s functionality from a user’s perspective. Use case analysis is one of the major activities of system requirements analysis and forms the backbone of the unified software development process (Jacobson, Booch, & Rumbaugh, 1999). A well-defined use case view is not only necessary for subsequent correct system design and implementation but also serves as a basis for future system evolution. Currently, only informal semantics of UML use case diagrams exists (UML, 2003). A use case diagram depicts the relationships between use cases and actors as well as relationships between use cases. There is no standard language for defining use cases, although a variety of choices, including plain text, state machines, operations, and interaction diagrams, is suggested; hence, informal plain text is often used in defining use cases. The lack of a precise standard language for defining use cases and their relationships makes the understanding and realization of use case diagrams difficult and the formal analysis of use case diagrams impossible. As a result, many errors due to incomplete and inconsistent requirements as well as incorrect specification cannot be detected and revealed until late in the system development process.

Hierarchical predicate transition nets (HPrTNs) are chosen to provide a formal semantics for UML use case diagrams for the following reasons: (1) HPrTNs (He, 1996) are a