Chapter XIII

Linking UML with Integrated Formal Techniques

Jing Liu, Jin Song Dong and Kun Shi
National University of Singapore

Brendan Mahony
Defence Science and Technology Organization, Australia

The challenge for complex systems specification is how to visually and precisely capture static, dynamic and real-time system properties in a highly structured way. In particular, requirement specifications for composite systems often involve capturing concurrent interactions between software control parts and physical system components/devices. The requirement specifications of such systems need to capture the structure and behavior of each individual physical/software components and their communications. In this chapter, we investigate the links between the graphical notation UML and an integrated formal notation. We present an effective combination of UML and an integrated formal method for the requirement specification of a light control system.

This work is supported in part by the research grant (Integrated Formal Methods) from National University of Singapore (No. RP3991615).

INTRODUCTION

Requirements capture is a key activity in software and system engineering. The challenge for complex system requirement specification is how to precisely capture static, dynamic and real-time system properties in a highly structured way. In particular, composite systems (Feather, 1987) often involve concurrent interactions between software control parts and physical system components/devices. The requirement specifications of such systems need to capture the structure and behavior of physical/software components and their communications. Formal methods are well known for their precision and expressiveness in specifying software and system requirements (Crow & Vito, 1998, Dandenne, Lamsweerde & Fickas, 1993; Dubois, Yu & Petit, 1998; Hesketh et al., 1998; Leveson et
However, formal specification techniques are not well integrated with existing industrial requirement analysis practices (Darimont, Heisel & Souquieres, 1999) and have a significant barrier to entry. On the other hand, though graphical notations are easy to adopt, they lack formal semantics and have scale up problems. Combinations of the two have been found successful (Grimm, 1998). To fight complexity, structured techniques (for example object-oriented methods) are also needed to properly partition the model into manageable individual components.

Following the success of conceptual modeling in databases (e.g., ER [Chen, 1976] and NIAM [Nijssen & Halpin, 1989]), many object-oriented modeling methods have been developed during the 1990s. Now those methods and notations have been merged into, the Unified Modeling Language (UML) (Rumbaugh, Jacobson & Booch, 1999). UML consists of various graphical notations which can capture the static system structure (class diagram), system component behavior (statechart diagram), system component interaction (collaboration and sequence diagram). The shortcomings of UML are:

- There is no unified formal semantics for all those diagrams. There are a few approaches to formalize a subset of UML, e.g., Evans and Clark (1998) and Kim and Carrington (1999) concentrated on class diagram semantics. Therefore, the consistency between diagrams is problematic; and
- There are limited capabilities for precisely modeling timed concurrency. For example, (in a new feature that has been added to the UML 1.3) synchronization between concurrent substates of a single statechart diagram can be captured using a synch state link. However, there is no facility to precisely model synchronous interactions between states in two different statechart diagrams.

If UML is combined with formal specification techniques, then its power can be further realized and enhanced. We believe that the best companions for UML are likely to be formal object-oriented methods. The two techniques are highly compatible and transparent to each other. One integrated formal object-oriented specification language is Timed Communicating Object Z (TCOZ) (Mahony & Dong, 1998). TCOZ combines the strengths of Object_Z (Duke, Rose & Smith, 1995) in modeling complex data and state with the strengths of Timed CSP (Schneider & Davies, 1995) in modeling real-time concurrency. In addition to CSP’s channel-based communication mechanism, in which messages represent discrete synchronizations between processes, TCOZ is extended with continuous function interface mechanisms inspired by process control theory, the sensors and the actuators (Mahony & Dong, 1999). Therefore, a combination of UML and TCOZ would be a good solution to the requirement specification of timed reactive control systems.

The key technique ideas in this approach are:

- Syntactically, UML/OCL (Object Constraint Language) is extended with TCOZ communication interface types — chan, sensor and actuator. Upon that, TCOZ sub-expressions can be used (in the same role as OCL) in the statechart diagrams and collaboration diagrams.
- Semantically, UML class diagrams are identified with the signatures of the TCOZ classes. The states of the UML statechart are identified with the TCOZ processes (operations) and the state transition links are identified with TCOZ events/guards.
- Effectively, UML diagrams can be seen as the viewpoint visual projections from a unified formal TCOZ model.

In this chapter, we illustrate such an effective combination of UML and TCOZ through the requirement specification of a light control system (LCS). The specification/design processes of this approach are:
Related Content

An Interactive Viewpoint on the Role of UML

Data Integration Issues and Opportunities in Biological XML Data Management

UML- and XML-Based Change Process and Data Model Definition for Product Evolution
Ping Jiang, Quentin Mair, Julian Newman and Josie Huang (2005). Software Evolution with UML and XML (pp. 190-221). www.irma-international.org/chapter/uml-xml-based-change-process/29614/

Extension of the Unified Modeling Language for Mobile Agents

A Framework for Cost-Based Query Optimization in Native XML Database Management Systems