Chapter 11

A Design Method for Real-Time Object-Oriented Systems Using Communicating Real-Time State Machines

Eduardo B. Fernandez, Jie Wu, and Debera R. Hancock
Florida Atlantic University, USA

Many methodologies for software modeling and design include some form of static and dynamic modeling to describe the structural and behavioral views respectively. Modeling and design of complex real-time software systems requires notations for describing concurrency, asynchronous event handling, communication between independent machines, timing properties. Dynamic modeling of real time systems using object-oriented methodologies requires extensions to the traditional state machine notations in order to convey the real-time system characteristics and constraints. This chapter proposes an object-oriented analysis and design methodology that augments the traditional UML (Unified Modeling Language) dynamic model with real-time extensions based on high-level parallel machines and communication notations from CRSM (Communicating Real-Time State Machines). An example of the proposed methodology is provided using a realistic example of an automated passenger train system.

INTRODUCTION

Real-time systems are characterized by their response requirements (deadlines) and underlying concurrency of functions. These time-critical systems often have stringent safety requirements, necessitating that they be highly reliable and
that their functions be predictable when subjected to real-time, concurrent events. Although the systems may be quite complex, good analysis and design methodologies must be simple and understandable while conveying accurately the design and its real-time aspects. Methodologies for specifying system requirements and designs typically include some form of requirements specification and notations for modeling the static, dynamic, and functional aspects of the system. This specification is often a textual description of the functional characteristics of the system and alone may not serve to accurately and unambiguously define the requirements. Formal languages are required to clearly specify and validate critical system requirements such as timing and safety constraints (Heitmeyer & Mandrioli, 1996). For real-time systems where there are often critical timing requirements or safety considerations, verification of the design and implementation with respect to formally specified critical requirements is necessary.

Once the requirements are specified, a series of analysis and design steps are performed that refine and map the requirements to a complete design. With object-oriented methods, the analysis step includes a representation of the real-world problem as a static class diagram. This modeling of the real-world problem into independent, data-encapsulated classes maps conceptually into the system as a collection of concurrently active communicating components. In a good methodology, the mapping of the problem information described in the requirements specification to the objects in the static class diagram should be consistent and visibly intuitive.

In practice, the complete system design is typically derived through a series of refinements from the static analysis model. It is often difficult to pinpoint when analysis ends and design starts. A series of iterations of lower level analysis and design steps is performed. Many notations and methodologies have been used for these steps. With any good methodology, some form of dynamic modeling is required to model and design the behavior of the system elements over time. State machines and variations on state machine notations are popular modeling tools for both object-oriented and functional methodologies. High-level state machines are often decomposed and refined into lower-level state machines. Sequence diagrams and scenario descriptions are examples of design approaches that are used to refine the operations and states of the lower level state machines.

Many real-time method extensions for function-oriented methodologies have been proposed and applied to real system development (Gomaa, 1986; Harel et al., 1990; Leveson & Heimdahl, 1994). Object-oriented design has recently become popular as an alternative for designing complex software systems. It is clear that the popularity and inherent concurrency of object-oriented designs makes it a highly desirable approach for producing real-time systems. Methodologies that map closely to object-oriented methodologies and programming languages have been developed such as O-Charts (Harel & Gary, 1996), Real-Time Object-
Related Content

On Temporal Summation in Chaotic Neural Network with Incremental Learning

Software System Modernization: An MDA-Based Approach
[www.igi-global.com/chapter/software-system-modernization/60721?camid=4v1a](www.igi-global.com/chapter/software-system-modernization/60721?camid=4v1a)

An Improved Model-Based Technique for Generating Test Scenarios from UML Class Diagrams
[www.igi-global.com/chapter/an-improved-model-based-technique-for-generating-test-scenarios-from-uml-class-diagrams/108629?camid=4v1a](www.igi-global.com/chapter/an-improved-model-based-technique-for-generating-test-scenarios-from-uml-class-diagrams/108629?camid=4v1a)
Information Theoretic XSS Attack Detection in Web Applications
Hossain Shahriar, Sarah North, Wei-Chuen Chen and Edward Mawangi (2014).
International Journal of Secure Software Engineering (pp. 1-15).
www.igi-global.com/article/information-theoretic-xss-attack-detection-in-web-applications/118145?camid=4v1a