Designing Concurrent, Distributed, and Real-Time Applications with UML

Hassan Gomaa
Department of Information and Software Engineering
George Mason University
Fairfax, Virginia 22030, USA
hgomaa@gmu.edu

Abstract

Object-oriented concepts are crucial in software design because they address fundamental issues of adaptation and evolution. With the proliferation of object-oriented notations and methods, the Unified Modeling Language (UML) has emerged to provide a standardized notation for describing object-oriented models. However, for the UML notation to be effectively applied, it needs to be used with an object-oriented analysis and design method. This tutorial describes the COMET method for designing real-time and distributed applications, which integrates object-oriented and concurrency concepts and uses UML.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques.

General Terms

Design, performance.

1. Introduction

Most books and courses on object-oriented analysis and design only address the design of sequential systems or omit the important design issues that need to be addressed when designing real-time and distributed applications [1]. It is essential to blend object-oriented concepts with the concepts of concurrent processing in order to successfully design these applications. As the UML is now the standardized notation for describing object-oriented models [2], this tutorial uses the UML notation throughout.

2. The COMET Method

COMET is a Concurrent Object Modeling and Architectural Design Method for the development of concurrent applications, in particular distributed and real-time applications [3]. The COMET Object-Oriented Software Life Cycle is highly iterative. In the Requirements Modeling phase, a use case model is developed in which the functional requirements of the system are defined in terms of actors and use cases.

In the Analysis Modeling phase, static and dynamic models of the system are developed. The static model defines the structural relationships among problem domain classes. Object structuring criteria are used to determine the objects to be considered for the analysis model. A dynamic model is then developed in which the use cases from the requirements model are refined to show the objects that participate in each use case and how they interact with each other. In the dynamic model, state dependent objects are defined using statecharts.

In the Design Modeling phase, an Architectural Design Model is developed. Subsystem structuring criteria are provided to design the overall software architecture. For distributed applications, a component based development approach is taken, in which each subsystem is designed as a distributed self-contained component. The emphasis is on the division of responsibility between clients and servers, including issues concerning the centralization vs. distribution of data and control, and the design of message communication interfaces, including synchronous, asynchronous, brokered, and group communication. Each concurrent subsystem is then designed, in terms of active objects (tasks) and passive objects. Task communication and synchronization interfaces are defined. The performance of real-time designs is estimated using an approach based on rate monotonic analysis [4].

Distinguishing features of the COMET method are:

1. Structuring criteria to assist the designer at different stages of the analysis and design process: subsystems, objects, and concurrent tasks.
2. Dynamic modeling, both object collaboration and statecharts, describing in detail how object collaborations and statecharts relate to each other.
3. Distributed application design, addressing the design of configurable distributed components and inter-component message communication interfaces.
4. Concurrent design, addressing in detail task structuring and the design of task interfaces.
5. Performance analysis of real-time designs using real-time scheduling.

COMET emphasizes the use of structuring criteria at certain stages in the analysis and design process. Object structuring criteria are used to help determine the objects in the system, subsystem structuring criteria are used to help determine the subsystems, and concurrent task structuring criteria are used to help determine the tasks (active objects) in the system. UML stereotypes [2] are used throughout to clearly show the use of the structuring criteria.

3. Requirements Modeling

In the requirements model, the system is considered as a black box. The use case model is developed.

- **Use Case Modeling.** Define actors and black box use cases. The functional requirements of the system are defined in terms of use cases and actors.

4. Analysis Modeling

In the analysis model, the emphasis is on understanding the problem; hence, the emphasis is on identifying the problem domain objects and the information passed between them. Issues such as whether the object is active or passive, whether the message sent is asynchronous or synchronous, and what operation is invoked at the receiving object are deferred until design time. The activities are

- **Static modeling.** Define problem-specific static model. This is a structural view of the information aspects of the system. Classes are defined in terms of their attributes, as well as their relationship with other classes. Operations are defined in the design model. The emphasis is on the information modeling of real-world classes in the problem domain.
- **Object structuring.** Determine the objects that participate in each use case. Object structuring criteria are provided to help determine the objects, which can be entity objects, interface objects, control objects, and application logic objects. After the objects have been determined, the dynamic relationships between objects are depicted in the dynamic model.
- **Finite State Machine Modeling.** The state-dependent aspects of the system are defined using hierarchical statecharts. Each state-dependent object is defined in terms of its constituent statechart.
- **Dynamic Modeling.** The use cases are refined to show the interaction among the objects participating in each use case. Collaboration diagrams or sequence diagrams are developed to show how objects collaborate with each other to execute the use case. The dynamic analysis approach is used to help determine how objects interact with each other to support the use cases. For state-dependent use cases, the interaction among the state-dependent control objects and the statecharts they execute needs to be explicitly modeled.

5. Design Modeling

In the design model, the solution domain is considered. During this phase, the analysis model is mapped to a concurrent design model. For concurrent applications, such as distributed and real-time applications, the following activities are performed:

- Make decisions about subsystem structure and interfaces. Develop the overall software architecture. Structure the application into subsystems.
- Make decisions about how to structure the distributed application into distributed subsystems, in which subsystems are designed as configurable components. Design the distributed software architecture by decomposing the system into distributed subsystems and defining the message communication interfaces between the subsystems.
- Make decisions about the characteristics of objects, in particular, whether they are active or passive. For each subsystem, structure the system into concurrent tasks (active objects). During task structuring, tasks are structured using the task structuring criteria, and task interfaces are defined. Make decisions about the characteristics of messages, in particular, whether they are asynchronous or synchronous.
- Make decisions about class interfaces. For each subsystem, design the information hiding classes (passive classes). Design the operations of each class and the parameters of each operation. Use inheritance to develop class hierarchies.
- Develop the detailed software design, addressing detailed issues concerning task synchronization and communication, and the internal design of concurrent tasks. Design connector classes.
- For real-time applications, analyze the performance of the design. Apply real-time scheduling to determine if the concurrent real-time design will meet performance goals. Investigate alternative software architectures.

6. References

1. H. Gomaa, “Software Design Methods for Concurrent and Real-Time Systems”, Addison Wesley, 1993.
2. G. Booch, J. Rumbaugh, I. Jacobson, “The Unified Modeling Language User Guide”, Addison Wesley, Reading MA, 1999.
3. H. Gomaa, “Designing Concurrent, Distributed, and Real-Time Applications with UML”, Addison Wesley Object Technology Series, Reading MA, 2000.
4. SEI, Carnegie Mellon University, “A Practioner's Handbook for Real-Time Systems”, Kluwer, 1993.