Modeling Information Systems in UML

Peter Rittgen
University College of Borås, Sweden

INTRODUCTION

The first approaches to object-oriented modeling appeared by the second half of the 1970s, but not much happened for more than a decade, so there were still barely more than a handful of modeling languages at the end of the 1980s. It was the early 1990s that witnessed an ever-growing market in competing object-oriented methods so that potential users found it increasingly difficult to identify any single method that suited their needs. This phenomenon came to be known as the "method wars." Towards the end of 1994 two of the "big" players, Grady Booch and Jim Rumbaugh, decided to join forces by integrating their respective approaches, the Booch method and OMT (Object Modeling Technique). In late 1995, Ivar Jacobson became a member of this team merging in his OOSE method (Object-Oriented Software Engineering). The efforts of the "three amigos" aimed at overcoming unnecessary differences between the individual approaches and also improving each of them by creating a common, standardized modeling language that could serve as an industry standard. The result was the release of the Unified Modeling Language (UML), version 0.9, in June 1996. The UML partners, an industry consortium, performed further work on UML. This led to the versions 1.0 and 1.1 being introduced in 1997. The latter was adopted by the OMG (Object Management Group) in the same year. The current version is 1.5 (OMG, 2003) but a major upgrade to 2.0 is in preparation (Björkander & Kobryn, 2003).

BACKGROUND

UML is a language to support the development of software systems. It features a common framework, which provides a set of diagram types covering different aspects of an information system. Here we will focus on the ones we deem to be most important: class diagram, use case diagram, and activity diagram. The first is the principal diagram for the static view on a system. The second is often put forward as the central diagram in communication with the user (see, e.g., Dobing & Parsons, 2000). The latter plays a central role in the information system (or business-oriented) perspective (see following sections). Elementary concepts of the UML are actor, activity (and state), object, and class.

An **actor** is a human being or a (computer) system who/which is able to perform activities on his/her/its own. The actor typically represents a group of similar human beings/systems and corresponds to the role the actor performs in the given context. *Teacher* is an example for such an actor. In UML actors are shown as stick figures.

An **activity** refers to a logically connected set of actions that are carried out as a unit in some order. These actions might be executed sequentially, alternatively, concurrently or in any combination thereof. *Grade exam* is an example for an activity.

*"An **object** is an abstraction of a set of real-world things such that:*

- *all of the real-world things in the set - the instances - have the same characteristics,*
- *all instances are subject to and conform to the same rules" (Shlaer & Mellor, 1988, p. 14).*

The structure (attributes) and behavior (operations) of similar objects are gathered in their common **class**. The values of the attributes at a certain point in time represent the state of the object. Classes are drawn as rectangles with the object identifier printed in bold face. Additional horizontal compartments can be introduced for the attributes and operations of the class. The object is depicted in the same way with the object identifier underlined (optionally followed by a colon and the respective class identifier). *Grade* is an example of a class, *F: Grade* that of an object.

Figure 1 shows how these concepts are related to each other: actors perform activities that involve objects. Each object is an instance of precisely one class. The class diagram shows classes and their relations (called associations). An activity diagram gives a detailed account of the order in which activities are executed. It can also show the relevant states of the involved objects. The use case diagram visualizes use cases (complex activities) and their relation to actors.

Figure 1. Language overview

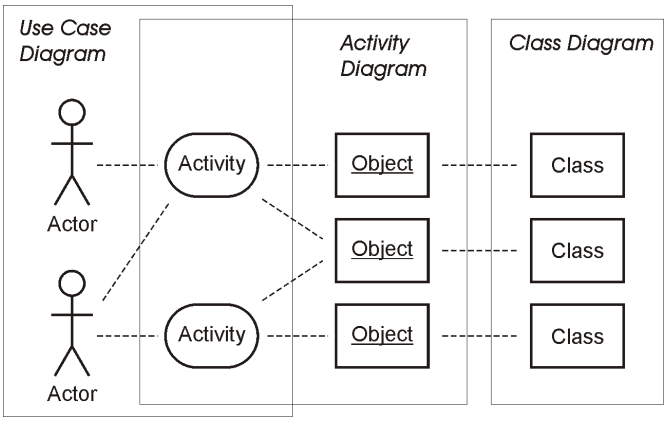
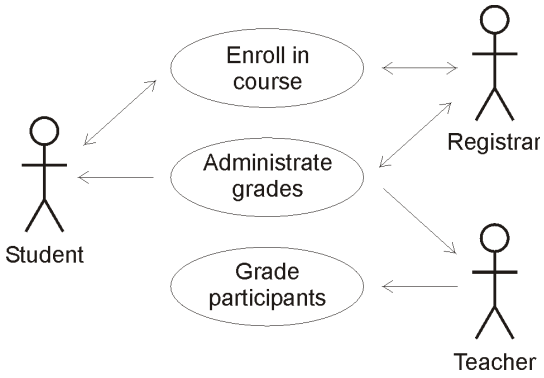


Figure 2. Use case diagram



Use Case Diagram

Use cases were introduced in 1992 by Jacobson and his colleagues. Their book is now available in the 2nd edition (Christerson, Jonsson, & Övergaard, 2004). A use case describes a way in which an actor can interact with your organization. It forms part of a use case scenario, a business situation that is supposed to be supported by the information system in question. *Figure 2* gives an example of a use case diagram involving 3 use cases and 3 actors.

The arrows indicate the direction of the information flow. So the registrar can both read and enter/change grades, whereas teachers and students can only read them. Often the arrowheads are omitted to simplify modeling. Each use case is detailed by a description in structured English of the activities that make up this use case. The use case diagram and the use cases form the basis for the development of class diagrams and activity diagrams. The former specify the static structure of the information that is handled by the use cases, and the latter give a precise, formalized account of the process logic.

Activity Diagram

An activity diagram consists of the detailed activities that make up the use cases. *Figure 3* gives an example of such a diagram in relation to the use cases of *Figure 2*.

The use case *Enroll in course* comprises the activities *Apply for course* (performed by the student) and *Enroll student* (performed by the registrar). *Grade participants* maps to *Grade exam* assuming that the grade for the course is determined by a final exam only. The use case *Administrate grades* involves the activities *Enter grade*, *Check grade* and *Change grade*. Note that the activity diagram also contains activities that are not present in the use case diagram, such as *Deliver course* and *Write exam*. That is because, in our example, they are not supposed to be supported by the information system and hence do not constitute use cases of such a system. But they are still an important part of the overall business process.

The business process in *Figure 3* starts with the initial state (the black dot). The activities are in the boxes with round sides. The rectangles contain objects-in-state where the object identifier is underlined and the object's state is enclosed in square brackets. So after *Enter grade*, for example, the object *Grade* is in state *entered*. Alternative paths leaving an activity must be labeled by so-called guards that determine under which condition each path is taken. These guards are also enclosed in square brackets and should be mutually exclusive. The process terminates when the final state is reached (i.e., the circle containing the black dot). Actors are included in the form of so-called