In this chapter and the next, we'll introduce you to the various UML models that you can create in VEA 2003. We'll show you how to create the UML models in VEA and provide you with a description of each model, along with tips for when to use it.

In this chapter, we'll concentrate on the contents of the model, focusing on how to create the models from a UML point of view. Our intention isn't to provide you with a complete guide on how to use Visio or a complete guide to UML. The goal is to introduce you to the models and elements you'll need when developing enterprise solutions.

VEA supports the eight models included in UML version 1.2 (http://www.omg.org/uml). Generally, the eight UML models can be divided into two categories:

- **Behavioral** (or **dynamic**) models, which include use case diagrams, sequence diagrams, collaboration diagrams, statechart diagrams, and activity diagrams

- **Static models**, which include static structure diagrams, component diagrams, and deployment diagrams

You use the behavioral models to document and specify what goes on in your solution; they describe the flow of operations and the events that take place. You use the static models to document and specify the static structure of your solution. You can compare this to building a house, where the static structure is the walls, floors, doors, and so on. The behavioral models would describe what goes on in the house—for example, the traffic through the rooms and so on.

VEA and IBM Rational XDE currently don't support code generation from behavioral diagrams;¹ you can generate code only from a static structure diagram (a class diagram).

---

¹ This doesn't seem likely to change in the near future.

J. E. Hansen et al., *Enterprise Development with Visual Studio .NET, UML, and MSF*  
© John Erik Hansen and Carsten Thomsen 2004
In this chapter, we'll concentrate on the behavioral models. The behavioral models are usually the first diagrams to be created, because you need to capture the requirements and the flows through the system before you can develop the solution architecture.

Statechart diagrams are created in order to model an object's life cycle. You do this by specifying which states your object can be in and which transitions, caused by outside stimuli, make the object transform from one state to another. Statechart diagrams are related to a class or a use case, which also means it's possible to base a statechart diagram on a use case. Basically, you can base a statechart diagram on almost any UML element, but in general, it's only done for some classes (those with interesting behavior). Because of this, we'll cover statechart diagrams together with class diagrams in the next chapter.

Use Case Diagrams

As you learned in previous chapters, use case diagrams are used for capturing user requirements. They are the first diagrams to be created. Why is this so? Well, it's because the use case diagrams describe the user requirements in a high-level, nontechnical manner. This conceptual design is crucial to the project's success. As you learned in Chapter 2, it's important to be confident that you know about the requirements that are likely to affect your choices before you decide on which technologies to use. The high-level nature of use case diagrams is excellent for seeing the project scope. These diagrams become the basis for the detailed models created later on as the project evolves.

Another use case diagram strength is that it uses plain text and simple drawings, without using any technical terms. As a result, use case diagrams are understandable to a broad audience.

Use case diagrams consist of the following basic items:

- Actors
- Relationships
- Processes
- Packages
- System boundaries

The following sections describe each of these elements.