Part II Modeling with Objects

Modeling is the central skill of analysis and design. It is how we describe the structure and behavior of things either as they exist or as we intend to build them. The important feature of a model is that it is a simplified description, showing just enough detail for the purpose at hand.

Models can be divided into three parts: static, dynamic, and interactive.

- The static part deals with the information we have about the state of an object at any given moment. At a given level of time granularity, we describe static attributes, relationships, and constraints between objects. Chapter 2, Static Models: Object Attributes and Invariants, is about modeling static aspects using abstract attributes.
- The dynamic part deals with the changes that happen to the state as events occur. It shows how actions affect objects, using the objects and attributes defined in the model of the object state. Chapter 3, Behavior Models: Object Types and Operations, is about describing these dynamic aspects of an object by specifying its operations in terms of effects on its attributes and information exchanged.
- The interactive part deals with interactions between objects. It shows how the responsibility for achieving a goal is divided among collaborating objects and how object interactions can be abstractly described. Chapter 4, Interaction Models: Use Cases, Actions, and Collaborations, is about describing and abstracting object interactions using use cases, joint actions, and collaborations.

Chapter 5, Effective Documentation, provides guidelines for documenting using these modeling techniques to aid in structuring the documentation and accompanying models.

If there's one thing that distinguishes object-oriented (OO) design methods from their predecessors, it is the Golden Rule that an object-oriented design is based on a model of the domain in which it works. Clearly, an early step in doing an OO design must therefore be to establish the objects that exist in the domain. Moreover, the language we use to describe the domain (the concepts and entities in the business with which the software is concerned) must be the same as the language for describing software designs. And we must have systematic ways of carrying this principle through to our designs and code. This is part of the power of an object technique: it allows hardware, software, and users, at all levels from business to code, to be seen as part of the same continuum of interacting objects.

Part II is mainly about techniques and notations, the language for modeling using objects. These techniques are used to describe how an individual object behaves externally (its specification as a type) and how it is designed internally as a group of interacting objects (a collaboration).

Objects that have similar behaviors are members of the same type; they satisfy the specification of that type. Behaviors are specified in terms of attributes that are a valid abstract model, called a *type model*, of many possible implementations. Each action is described in terms of its effect on the attributes of the participating objects and the outputs it produces. The most interesting aspects of a design are the interactions between objects. You can abstract away detailed interaction protocols between objects by using joint actions and collaborations; and you can describe specific interactions as refinements of a more abstract description.

These techniques apply at the level of a business process, problem domain description, software component specification, design, or implementation. Part V describes how to apply these techniques at different levels in a systematic process. It says more about the process of discovering objects and of achieving continuity and traceability from problem domain to code.