2

Modeling as a Design Technique

A ***model*** is an abstraction of something for the purpose of understanding it before building it. Because a model omits nonessential details, it is easier to manipulate than the original entity. Abstraction is a fundamental human capability that permits us to deal with complexity. Engineers, artists, and craftsmen have built models for thousands of years to try out designs before executing them. Development of hardware and software systems is no exception. To build complex systems, the developer must abstract different views of the system, build models using precise notations, verify that the models satisfy the requirements of the system, and gradually add detail to transform the models into an implementation.

2.1 Modeling

Designers build many kinds of models for various purposes before constructing things. Examples include architectural models to show customers, airplane scale models for wind-tunnel tests, pencil sketches for composition of oil paintings, blueprints of machine parts, storyboards of advertisements, and outlines of books. Models serve several purposes.

- **Testing a physical entity before building it**. The medieval masons did not know modern physics, but they built scale models of the Gothic cathedrals to test the forces on the structure. Engineers test scale models of airplanes, cars, and boats in wind tunnels and water tanks to improve their dynamics. Recent advances in computation permit the simulation of many physical structures without the need to build physical models. Not only is simulation cheaper, but it provides information that is too fleeting or inaccessible to be measured from a physical model. Both physical models and computer models are usually cheaper than building a complete system and enable early correction of flaws.
- **Communication with customers**. Architects and product designers build models to show their customers. Mock-ups are demonstration products that imitate some or all of the external behavior of a system.

- **Visualization**. Storyboards of movies, television shows, and advertisements let writers see how their ideas flow. They can modify awkward transitions, dangling ends, and unnecessary segments before detailed writing begins. Artists' sketches let them block out their ideas and make changes before committing them to oil or stone.
- **Reduction of complexity**. Perhaps the main reason for modeling, which incorporates all the previous reasons, is to deal with systems that are too complex to understand directly. The human mind can cope with only a limited amount of information at one time. Models reduce complexity by separating out a small number of important things to deal with at a time.

2.2 Abstraction

Abstraction is the selective examination of certain aspects of a problem. The goal of abstraction is to isolate those aspects that are important for some purpose and suppress those aspects that are unimportant. Abstraction must always be for some purpose, because the purpose determines what is, and is not, important. Many different abstractions of the same thing are possible, depending on the purpose for which they are made.

All abstractions are incomplete and inaccurate. Reality is a seamless web. Anything we say about it, any description of it, is an abridgement. All human words and language are abstractions—incomplete descriptions of the real world. This does not destroy their usefulness. The purpose of an abstraction is to limit the universe so we can understand. In building models, therefore, you must not search for absolute truth but for adequacy for some purpose. There is no single "correct" model of a situation, only adequate and inadequate ones.

A good model captures the crucial aspects of a problem and omits the others. Most computer languages, for example, are poor vehicles for modeling algorithms because they force the specification of implementation details that are irrelevant to the algorithm. A model that contains extraneous detail unnecessarily limits your choice of design decisions and diverts attention from the real issues.

2.3 The Three Models

We find it useful to model a system from three related but different viewpoints, each capturing important aspects of the system, but all required for a complete description. The *class model* represents the static, structural, "data" aspects of a system. The *state model* represents the temporal, behavioral, "control" aspects of a system. The *interaction model* represents the collaboration of individual objects, the "interaction" aspects of a system. A typical software procedure incorporates all three aspects: It uses data structures (class model), it sequences operations in time (state model), and it passes data and control among objects (interaction model). Each model contains references to entities in other models. For example, the class model attaches operations to classes, while the state and interaction models elaborate the operations.

The three kinds of models separate a system into distinct views. The different models are not completely independent—a system is more than a collection of independent parts—but each model can be examined and understood by itself to a large extent. The different models have limited and explicit interconnections. Of course, it is always possible to create bad designs in which the three models are so intertwined that they cannot be separated, but a good design isolates the different aspects of a system and limits the coupling between them.

Each of the three models evolves during development. First analysts construct a model of the application without regard for eventual implementation. Then designers add solution constructs to the model. Implementers code both application and solution constructs. The word *model* has two dimensions—a view of a system (class model, state model, or interaction model) and a stage of development (analysis, design, or implementation). The meaning is generally clear from context.

2.3.1 Class Model

The ***class model*** describes the structure of objects in a system—their identity, their relationships to other objects, their attributes, and their operations. The class model provides context for the state and interaction models. Changes and interactions are meaningless unless there is something to be changed or with which to interact. Objects are the units into which we divide the world, the molecules of our models.

Our goal in constructing a class model is to capture those concepts from the real world that are important to an application. In modeling an engineering problem, the class model should contain terms familiar to engineers; in modeling a business problem, terms from the business; in modeling a user interface, terms from the application. An analysis model should not contain computer constructs unless the application being modeled is inherently a computer problem, such as a compiler or an operating system. The design model describes how to solve a problem and may contain computer constructs.

Class diagrams express the class model. Generalization lets classes share structure and behavior, and associations relate the classes. Classes define the attribute values carried by each object and the operations that each object performs or undergoes.

2.3.2 State Model

The ***state model*** describes those aspects of objects concerned with time and the sequencing of operations—events that mark changes, states that define the context for events, and the organization of events and states. The state model captures ***control,*** the aspect of a system that describes the sequences of operations that occur, without regard for what the operations do, what they operate on, or how they are implemented.

State diagrams express the state model. Each state diagram shows the state and event sequences permitted in a system for one class of objects. State diagrams refer to the other models. Actions and events in a state diagram become operations on objects in the class model. References between state diagrams become interactions in the interaction model.

2.3.3 Interaction Model

The ***interaction model*** describes interactions between objects—how individual objects collaborate to achieve the behavior of the system as a whole. The state and interaction models describe different aspects of behavior, and you need both to describe behavior fully.

Use cases, sequence diagrams, and activity diagrams document the interaction model. Use cases document major themes for interaction between the system and outside actors. Sequence diagrams show the objects that interact and the time sequence of their interactions. Activity diagrams show the flow of control among the processing steps of a computation.

2.3.4 Relationship Among the Models

Each model describes one aspect of the system but contains references to the other models. The class model describes data structure on which the state and interaction models operate. The operations in the class model correspond to events and actions. The state model describes the control structure of objects. It shows decisions that depend on object values and causes actions that change object values and state. The interaction model focuses on the exchanges between objects and provides a holistic overview of the operation of a system.

There are occasional ambiguities about which model should contain a piece of information. This is natural, because any abstraction is only a rough cut at reality; something will inevitably straddle the boundaries. Some properties of a system may be poorly represented by the models. This is also normal, because no abstraction is perfect; the goal is to simplify the system description without loading down the model with so many constructs that it becomes a burden and not a help. For those things that the model does not adequately capture, natural language or application-specific notation is still perfectly acceptable.

2.4 Chapter Summary

Models are abstractions built to understand a problem before implementing a solution. All abstractions are subsets of reality selected for a particular purpose.

We recommend three kinds of models. The class model describes the static structure of a system in terms of classes and relationships. The state model describes the control structure of a system in terms of events and states. The interaction model describes how individual objects collaborate to achieve the behavior of the system as a whole. Different problems place different emphasis on the three kinds of models.

abstraction	modeling
class model	relationship among models
interaction model	state model

Figure 2.1 Key concepts for Chapter 2