Using state diagrams in system development

State diagrams model the system from the point of view of a single class and the events that can affect the objects of the class. They show all possible behaviours of objects of a class, and record the ordering of events, for example in the Wheels system a bike must be assigned a number before it can be hired. This information about timing constraints is vital for our understanding of the system, and is not recorded in any of the other system models that we cover in this book.

Although state diagrams are a very useful and important modelling technique, it is not necessary to draw one for every class. In most systems, complexity arises from interaction between objects of different classes, as modelled in sequence and collaboration diagrams (see Chapter 6). It may well be that in any system, particularly an information system, only a few classes will display dynamic behaviour and so need a state diagram to model what happens. This is the case with a system such as Wheels, where most classes have objects that undergo only a restricted set of events and all the objects respond to the same event in the same way. For example, all Customer objects in the Wheels case study system respond to events that happen to them in the same way, although with different values: recording details, finding an associated Hire object, displaying customer details and amending customer details. Their response to events does not depend on what state they are in. For this sort of class which has relatively simple behaviour there is not a lot to show on a state diagram. However, other types of computer system, for example process control or communication systems, frequently have a number of classes whose dynamic behaviour is extremely complex. For this sort of class it is important to document all the possible behaviours of the objects of the class by means of a state diagram.

As with all models that are produced as part of the development process, it is important to check that state diagrams are consistent with other diagrams. A state diagram of a particular class should be checked against interaction diagrams which involve objects of the class to ensure that all the events in the state diagram appear in the interaction diagram as an incoming message to the object. It should also be checked against the class diagram to make certain that every event and action corresponds to an operation on the relevant class.

Technical points

Different types of event. Not all events occur because of outside influences. For example, an event can happen in the course of time. In the state diagram, this is represented by the keyword 'after', for