Generalization. Both use cases and actors can be specialized, i.e. we can use an inheritance (otherwise known as a generalization/specialization) relationship between actors and between use cases. In object-oriented development inheritance is primarily associated with class diagrams (see section on inheritance in Chapter 4). If an inheritance relationship exists between two entities it means that one inherits the characteristics of the other. For example, in a classification of animals, a horse is a specialization of animal and a cart-horse is a specialization of horse. Each specialization both inherits characteristics from its parent in the hierarchy and adds features to reflect its own specialization. The parent has those features common to all of its descendants, those features they have in general – hence generalization.

Where use cases are concerned, the use of generalization is another way of specifying alternative courses of events. We could, for example, have modelled the situation shown in Figure 3.12 by using a generalization relationship as in Figure 3.13. This allows us to model what happens more precisely. If the Receptionist is issuing a bike to a completely new customer then she uses the specialized use case 'Issue bike to new customer'. This use case always executes the use case 'Maintain customer list'. However, if she is issuing a bike to an existing customer she uses the generalized use case 'Issue bike'. If she finds, in the course of executing this use case, that the customer has changed their address she can opt to execute the 'Maintain customer list' use case.

Actors can also be specialized if we wish to indicate that the specialized actor plays the same role as the generalized one but has some extra role.

Initiating actor and participating actor. It is worth understanding the differences between types of actors without getting too bogged down in the details. In any given use case, the initiating actor is the

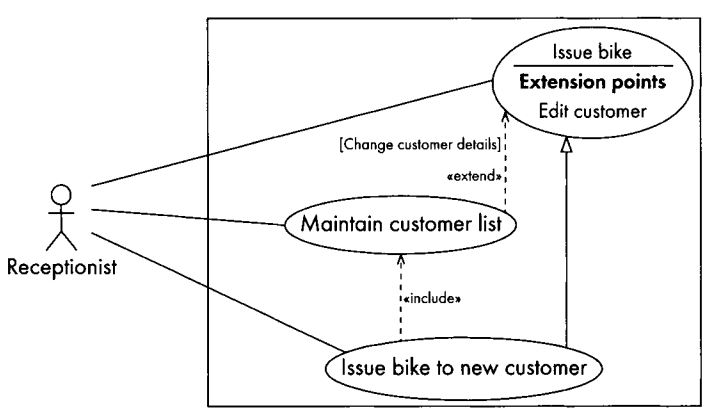

Figure 3.13 Generalization/specialization relationship