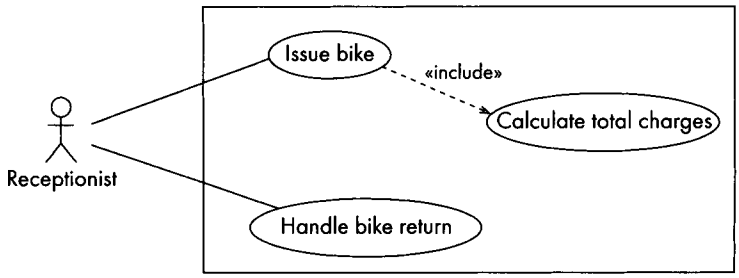

Figure 3.19 Unnecessary use of «include»

was sometimes part of the 'Issue bike' process, it was often something that was done on its own. Customers frequently phone up requiring information about the types of bike available and about the deposit and rental charges. For this reason we have modelled it as a separate use case.

From a practical point of view, the developer should beware of making use cases too small. Each use case spawns a large number of other models. Each use case will potentially have associated with it a use case description, a number of scenarios and several interaction diagrams. If each use case models only a small amount of functionality we need more of them to model the whole system and the number of models spawned increases exponentially.

3 How do I know when it is appropriate to use an «include» relationship?

It is very easy to make mistakes when using both «include» and «extend» relationships. One common error, on discovery of the technique, is to use it over enthusiastically. In Figure 3.19 a use case 'Calculate total charges' has been introduced and linked to 'Issue bike'. 'Calculate total charges' works out the rental charges for a bike for a given number of days and adds it to the deposit. The point of using «include» is to save effort by identifying behaviour common to more than one use case. However, the behaviour specified in 'Calculate total charges' would only ever be used by the 'Issue bike' use case; it is therefore unnecessary to model it as a separate use case.

Chapter summary

Use cases model the user's view of the functionality of a system. The use case model consists of a use case diagram, supported by textual descriptions, use case and actor descriptions, and scenarios. Both the diagrams and the supporting text are simple and intuitive which makes them an ideal vehicle for discussions with the user