Figure 3.2). Communication associations tell us which actors are associated with which use cases. Each actor may be associated with many use cases and each use case may be associated with many actors.

Include. Two other types of relationship may be used on a use case diagram – «include» and «extend». Both «include» and «extend» are relationships between use cases. An «include» relationship is useful when you find you have a chunk of behaviour that is common to several use cases. Rather than repeat a description of that behaviour in several use case descriptions, the common behaviour can be split off into a separate use case which is then linked to all relevant use cases with an «include» relationship. This is a similar idea to the programming technique of using procedures or subroutines to code cohesive bits of functionality that may be used in more than one place in a program. It is important to identify functionality common to several use cases because it permits the developers to avoid wasting effort; a chunk of functionality can be identified, investigated and modelled once, then reused as required.

Figure 3.9 shows a refined version of the original Wheels use case diagram (see Figure 3.2). The use cases 'Maintain bike list', 'Handle enquiries', 'Issue bike' and 'Handle bike return' all need to find a particular bike from the list of bikes. We therefore create a new use case 'Find bike' and link it with an «include» relationship to the four use cases that need it. This tells us that each of these use cases will always use the 'Find bike' use case.

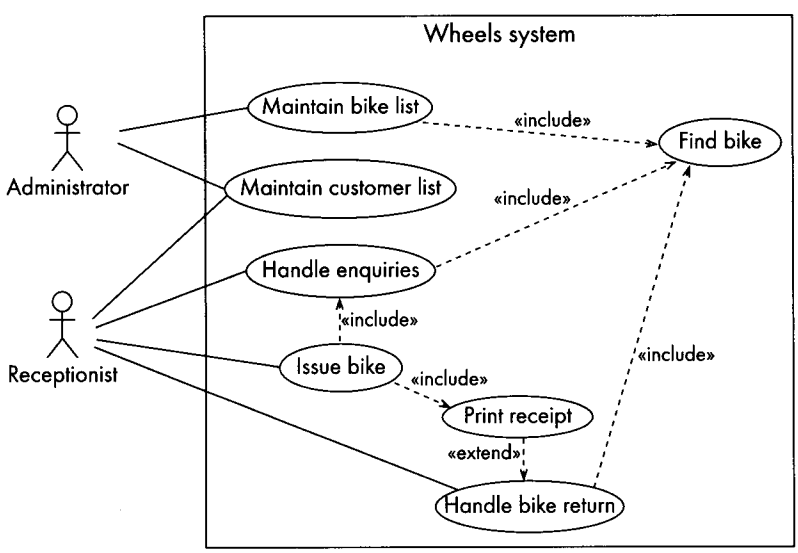

Figure 3.9 Wheels system with includes and extends relationships