Use case: Issue bike
Preconditions: 'Maintain bike list' must have been executed
Actors: Receptionist
Goal: To hire out a bike

Overview:
When a customer comes into the shop they choose a bike to hire. The Receptionist looks up the bike on the system and tells the customer how much it will cost to hire the bike for a specified period. The customer pays, is issued with a receipt, then leaves with the bike.

Cross-reference:
R3, R4, R5, R6, R7, R8, R9, R10

Typical course of events:

Actor action	System response
1 The customer chooses a bike	
2 The Receptionist keys in the bike number	3 **Initiate** 'Find bike'
4 Customer specifies length of hire	
5 Receptionist keys this in	6 Displays total hire cost
7 Customer agrees the price	
8 Receptionist keys in the customer identification	9 **Initiate** 'Maintain customer list'
10 Customer pays the total cost	
11 Receptionist records amount paid	12 **Initiate** 'Print receipt'

Alternative courses:

Steps 7–12	The customer may not be happy with the price and may terminate the transaction

Figure 3.11 *Expanded description of the 'Issue bike' use case documenting «include» and «extend» relationships*

as a line round the use cases, separating them from the actors; normally it is labelled with the name of the system or subsystem. The drawing of a boundary on a use case diagram, however, adds very little to the meaning of the diagram as use cases are always inside the boundary and the actors outside it. The reason we have included the boundaries on our diagrams is simply that it comes as part of the CASE tool we used to draw the diagrams, i.e. Together™. It is common practice, however, to omit the boundary and other CASE tools, for example Rational Rose™, do not draw a boundary on use case diagrams.