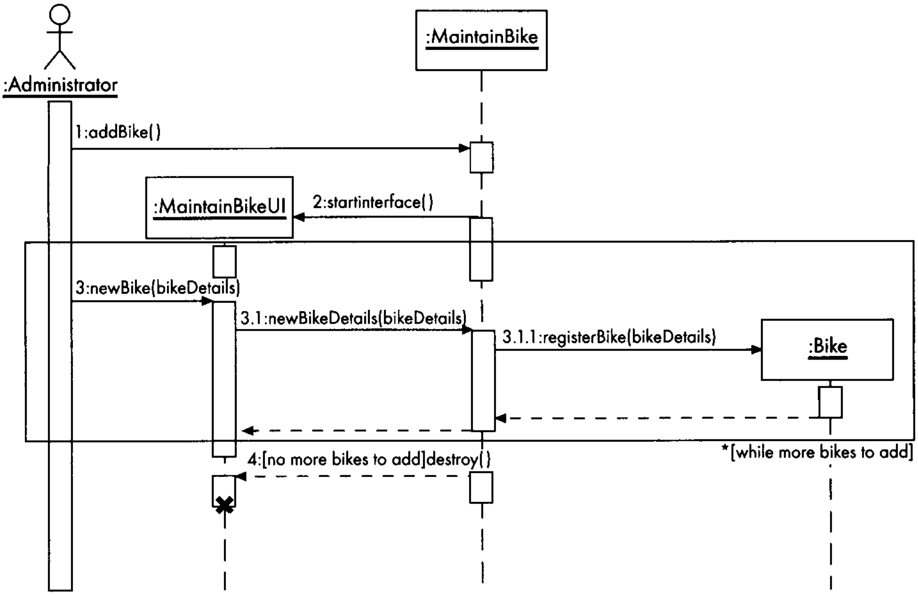
:Administrator
:MaintainBike
1:addBike()
:MaintainBikeUI
2:startinterface()
3:newBike(bikeDetails)
3.1:newBikeDetails(bikeDetails)
3.1.1:registerBike(bikeDetails)
:Bike
4:[no more bikes to add]destroy()
*[while more bikes to add]

Figure 10.8 Repeated block of code

evaluates to true. The UML does not dictate a format for writing conditions – natural language, structured English, pseudo-code, OCL (Object Constraint Language)[4] or a programming language expression are all acceptable.

Sequence diagrams allow us to model conditional branching, i.e. sending different messages depending on the value of a condition. Figure 10.9 shows a fragment of a sequence diagram from the Wheels system – for simplicity we have omitted interface and control objects. If a customer has not hired a bike before, the Receptionist must enter their details into the system. The Receptionist inputs the customer's name; if the system finds a :Customer with a matching name it displays the address. Otherwise a new :Customer is created. Only one of these messages should be sent, so it is important that the conditions should be mutually exclusive.

Showing conditional behaviour has the effect of modelling more than one possible sequence of events on one diagram. This means you can get away with drawing fewer sequence diagrams, but the disadvantage is that the diagrams become more complicated. If your diagram starts to get too complicated, show the different sequences of events on different diagrams.

[4] *The UML's formal language for specifying constraints.*