8 Activity diagrams

Learning outcomes

The material and exercises in this chapter will enable you to:

- Explain the role and purpose of activity diagrams in object-oriented systems development
- Draw a simple activity diagram
- Use activity diagrams to analyse business workflows, use cases and operations on classes.

Key words that you will find in the glossary:

- activity
- guard
- object flow
- synchronization bar
- fork
- join
- swimlane

Introduction

As we have seen in Chapter 3, the functionality of the system is initially represented through use cases, which describe the main activities of the system from the perspective of the user. System functionality is also specified in the operations on each class in the class diagram (see Chapters 5 and 6); the interaction diagrams (Chapter 6) specify the inter-object message passing required to achieve a particular task, and state diagrams (Chapter 7) model all possible behaviours of the objects of a class.

In this chapter we look at activity diagrams, which are used to model the details of complex processes. Activity diagrams are similar to state diagrams in that they are concerned with states and

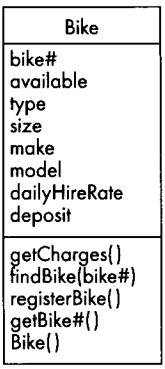


Figure 8.1 *The Bike class from the Wheels class diagram*

transitions between the states. However, in an activity diagram, all the states are activities (i.e. a state of doing something) and the transitions between them are triggered by the completion of the activity, rather than by an external event.

Activity diagrams show the internal flow of control in a process. They can be used to model processing at different levels, such as high-level workflows in an organization, the detail of what happens in a use case (as an alternative to a use case description), or they can specify in detail how an operation works (as an alternative to a process specification). Activity diagrams can be used to represent sequence, selection and iteration (structures that are found in nearly all programs) and they can also illustrate where different activities can be carried out in parallel.

Modelling a sequence of activities

The first example of an activity diagram is a simple model of the operation to calculate the amount to be paid when a bike is hired. Figure 8.1 shows the Bike class from the Wheels class diagram (for the complete diagram, see Figure 6.6 on page 155).

One of the operations on the Bike class is 'getCharges( )', but the class diagram only records the name of the operation. There are no details of what actually happens in the 'getCharges' operation. These can be specified in an operation specification (see Chapter 6) or an activity diagram.

Figure 8.2 shows an activity diagram illustrating the sequence of actions involved in the 'getCharges()' operation.

This is a very simple sequential diagram; most activity diagrams model more complex processing and use a fuller notation as shown in Figure 8.3.

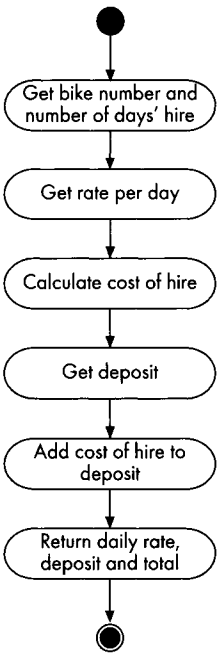


Figure 8.2 Simple activity diagram for the 'getCharges()' operation

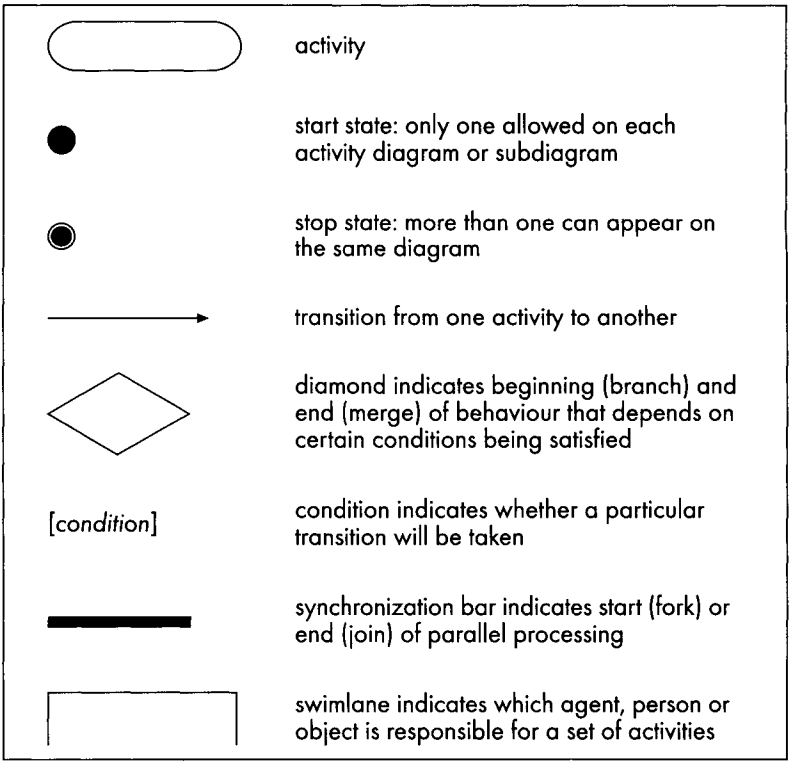


Figure 8.3 Activity diagram symbols

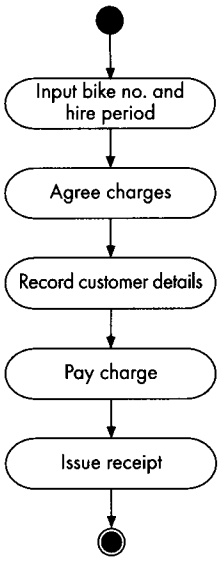


Figure 8.4 *Initial activity diagram for the 'Issue bike' use case*

Modelling alternative courses of action

One of the advantages of activity diagrams is that they can model different possible courses of action and the conditions which determine which course is taken.

Figure 8.4 shows an initial activity diagram for the 'Issue bike' use case.

Although this diagram illustrates the sequence of processing that occurs when a bike is issued, it only covers the situation where the customer is new to the Wheels system. In the case of an existing customer, it would be inefficient and confusing to input customer details each time the customer hires a bike; all the system needs to do is confirm that the customer details on record are correct.

Figure 8.5 shows an amended diagram for the 'Issue bike' use case that caters for both new and existing customers. The decision point is shown by the first diamond, and the conditions or guards for taking particular courses of action (whether the customer is new or existing) are indicated in square brackets.

As in state diagrams, it is essential that every guard evaluates to true or false, and that the guards on the alternative processing routes are mutually exclusive (for example, a customer cannot be both new and existing). This is to ensure that there is no ambiguity as to which route should be taken. The guard on one of the processing routes may be simply 'else', indicating that this is the default route in the case where all the other guards are false.

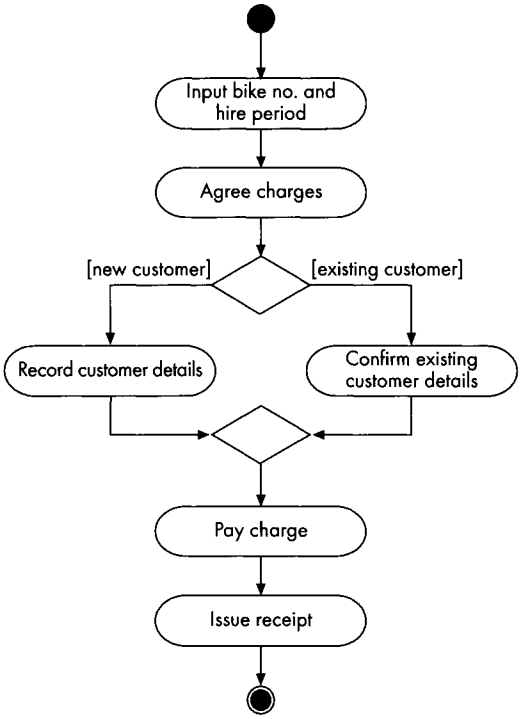


Figure 8.5 *Activity diagram for the 'Issue bike' use case, showing alternative actions*

Modelling iteration of activities

In addition to sequencing (see Figure 8.4) and selection (Figure 8.5) of activities, activity diagrams can also model iteration, where one or more activities need to be repeated. Figure 8.6 shows what happens when Naresh, the chief mechanic at Wheels, has to register a number of different bikes on the system. For each bike, Naresh has to enter the details and then assign a number; these activities are repeated until all the bikes have been registered. The diagram in Figure 8.6 shows the iteration loop, with the guard condition '[more bikes to add]' in square brackets.

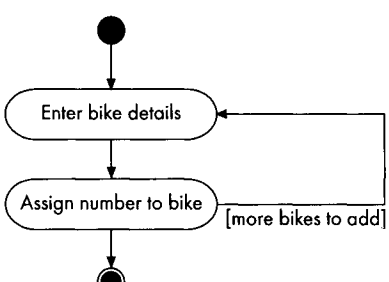


Figure 8.6 *Activity diagram showing iteration of activities*

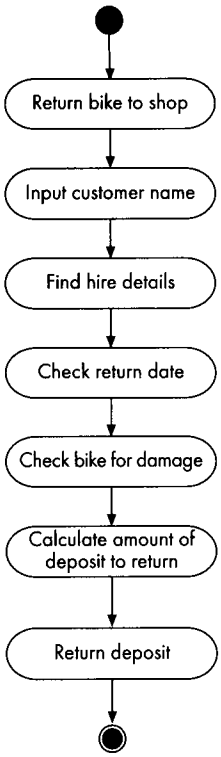


Figure 8.7 *Initial activity diagram for the 'Handle bike return' use case*

Modelling activities that are carried out in parallel

A further advantage of activity diagrams is that they illustrate where activities can be performed in parallel. In fact, the process of drawing an activity diagram often uncovers the possibility of performing in parallel activities that have previously been carried out sequentially. Figure 8.7 shows an initial activity diagram for the use case 'Handle bike return'.

We know from earlier investigations (see Chapter 2) that the order in which these activities are performed is irrelevant; the return date can be processed before checking the bike or vice versa. This means that the activities 'Check bike' and 'Check return date' can be shown on the activity diagram in parallel, as can be seen in the amended diagram in Figure 8.8.

In Figure 8.8 the top synchronization bar indicates that once the activity ('Find hire details') that is the source of the single incoming transition has completed, the outgoing transitions ('Check return date' and 'Check bike for damage') are taken in any order. The bottom synchronization bar indicates that the single outgoing transition is only triggered once both these activities have completed. A synchronization bar that signals the start of parallel activities is known as a fork, and one that signals the end of the

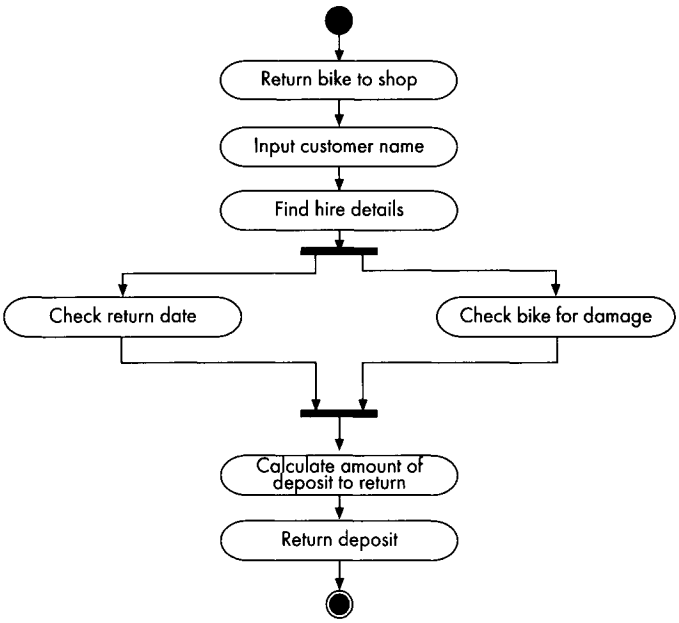


Figure 8.8 *Activity diagram for the 'Handle bike return' use case, showing parallel activities*

activities is known as a join. The transitions at the beginning and end of parallel activities must match; all outgoing transitions from the fork must eventually meet at the corresponding join.

Different types of activity structures, such as sequence, selection, iteration and parallel activities, can all occur in the same diagram, although this can sometimes make the diagram cluttered and difficult to read. Figure 8.9 shows a modified version of the activity diagram for the 'Handle bike return' use case, including selection and parallel activities. The diagram now shows what happens when a bike is overdue or returned damaged.

Swimlanes

None of the example activity diagrams shown so far in this chapter has given any indication of which person, agent or object carries out a given activity. Diagrams like these are actually very useful in the early stages of development when we want to think about what happens during processing without worrying about who or what has responsibility for a specific activity. Later on, however, it is useful in relation to each activity to be able to identify who, what, or which object in the system carries it out. We can add this information to an activity diagram by dividing the diagram into vertical zones, known as swimlanes. Swimlanes are separated from each other by lines and the top of each swimlane is labelled with

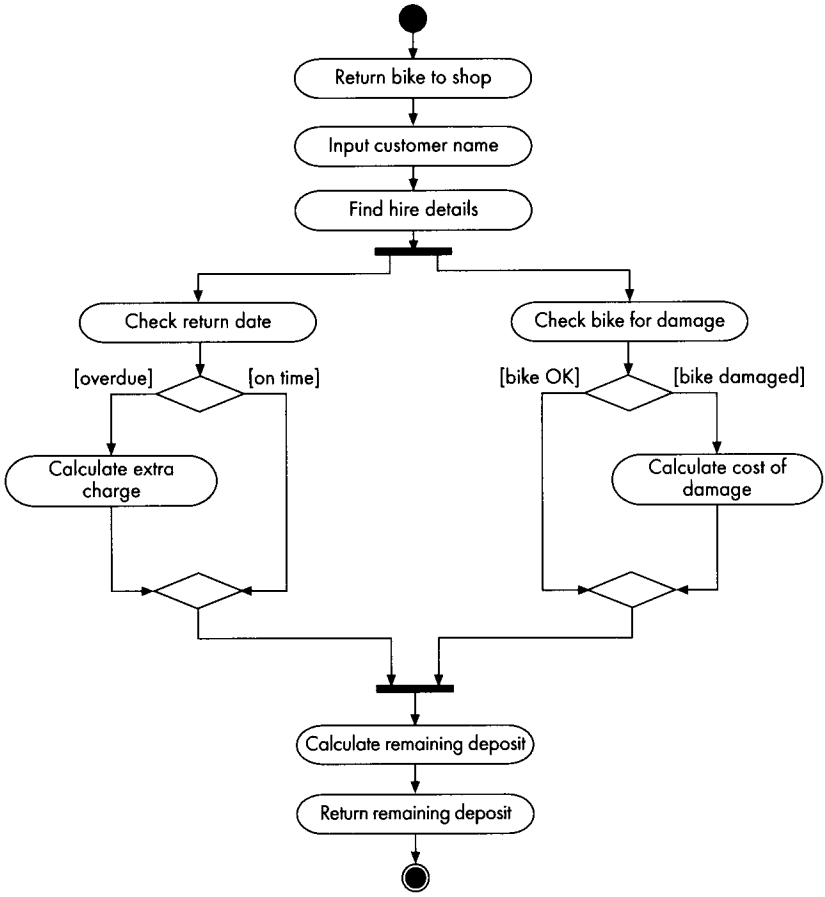


Figure 8.9 *Modified activity diagram for the 'Handle bike return' use case, showing parallel activities and selection*

the name of the person, organization or object responsible for carrying out the set of activities in the swimlane.

Figure 8.10 shows the activity diagram for the 'Handle bike return' use case (compare Figure 8.9). In Figure 8.10 swimlanes have been added to provide information about who carries out the various activities in the use case.

We can see from Figure 8.10 who or what carries out the different activities that make up the 'Handle bike return' use case. The customer is responsible for returning the bike to the shop. The receptionist inputs the customer name, and the computer carries out the activities of finding the hire details, checking the return date, calculating the extra charge if necessary, and calculating the remaining deposit. The mechanic is responsible for checking the bike for damage and calculating the cost of any damage discovered. Finally, the receptionist returns the remaining deposit to the customer.

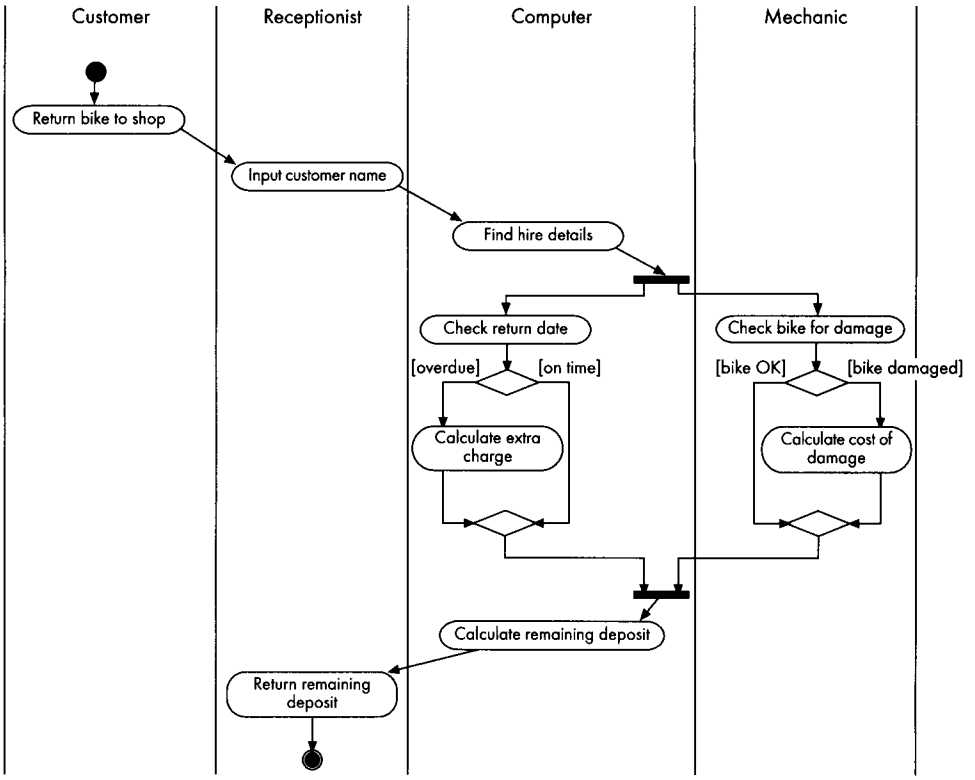


Figure 8.10 *Modified activity diagram for the 'Handle bike return' use case, showing responsibilities for different activities through the addition of swimlanes*

As we can see from the example, swimlanes are very useful for showing who does what in a workflow or use case, and how the processing in an operation is divided up between objects. However, the responsibilities of different objects can be seen more clearly in sequence and collaboration diagrams (see Chapter 6) and it is often better to omit extra details such as swimlanes from an activity diagram so that there is less risk of clutter. As with all diagrams, it is important to make sure that an activity diagram is easy to read.

Using activity diagrams in system development

Activity diagrams are a relatively recent addition to the UML, and many people dislike using them because they are process-based, rather than object-oriented. However, the diagrams are a useful and effective modelling tool that can be used throughout the system development process. They help to visualize the functionality of the system at different levels of detail, and aid communication between developers and clients. UML does provide text-based alternatives to activity diagrams, such as use case and

process descriptions, but clients generally find diagrammatic techniques, such as activity diagrams, easier to understand.

Activity diagrams can be drawn in the initial stages of development to help both developers and clients to analyse business workflow processes and gain a shared understanding of what is going on in the system. At this stage they provide a useful vehicle for discussion, helping developers, clients and users to visualize the system functionality.

The ability of activity diagrams to represent activities that can be carried out in parallel is particularly useful in high-level business modelling, as drawing the diagrams can help to identify potential for parallel processing, even where activities are currently carried out sequentially. Representation of parallel processing is especially useful in certain types of system, such as real time, where synchronization of activities and tasks is central to the system functionality.

Once the system use cases have been identified (see Chapter 3), activity diagrams can be used to illustrate the steps involved in achieving a use case goal, showing the activities and the order in which they take place.

Finally, when development has reached a stage where classes have been identified together with their attributes and operations, activity diagrams are a useful means of describing how the operations work, particularly when these are based on complex algorithms.

Technical points

Modelling iteration. When we discussed iteration in activity diagrams earlier in this chapter, we showed how to model it using a loop between activities (see Figure 8.6). It is also possible to show iteration using a multiplicity symbol * on an activity, which is useful when there is a risk of a diagram becoming cluttered. Figure 8.11 shows a section of an activity diagram illustrating what happens when Annie Price, the shop manager for Wheels, checks that the insurance on each bike is up to date. The multiplicity symbol on the activity 'Check bike insurance details' indicates that this activity is repeated until the details on all the bikes have been checked.

Omitting the diamond decision symbol. It is not mandatory to include the diamond symbol to indicate a decision leading to alternative courses of action in an activity diagram, although the different paths through the diagram are often clearer when the diamond is included. Figure 8.12 shows two versions of a section from Figure 8.5 (Activity diagram for the 'Issue bike' use case) with and without the diamond decision symbol.

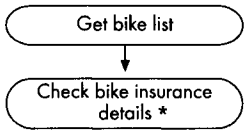
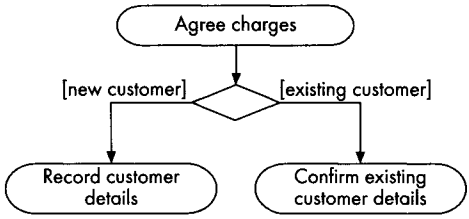


Figure 8.11 *Section of an activity diagram illustrating the use of the multiplicity symbol * to indicate repetition of an activity*



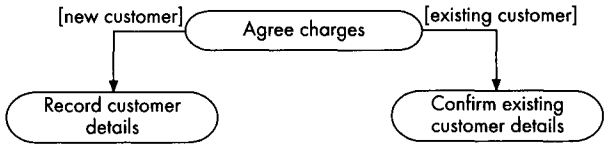


Figure 8.12 *Two versions of part of Figure 8.5, with and without the diamond decision symbol*

Partitioning the diagram. A subset of related activities on a diagram can be enclosed and labelled as shown in Figure 8.13, where the activities concerning the handling of customer details are represented as a subsection of the main diagram with its own start and stop states.

Partitioning an activity diagram in this way can help the readability of the overall diagram, and also supports reuse, since this subsection of the main diagram can be reused in any activity diagram which includes handling customer details.

Object flows. It is often useful to include in an activity diagram information about the input that an activity needs from a specific object, or how an object is affected by the output from an activity. In this way the processing represented in an activity diagram can be linked to its input and outputs.

Sometimes the name of an object is used as the name of an activity as in Figure 8.14.

Usually, however, the links between activities and objects are shown by including the relevant objects in the activity diagram, together with object flows to or from the associated activity. If an object provides input for an activity, an object flow (dashed arrow)

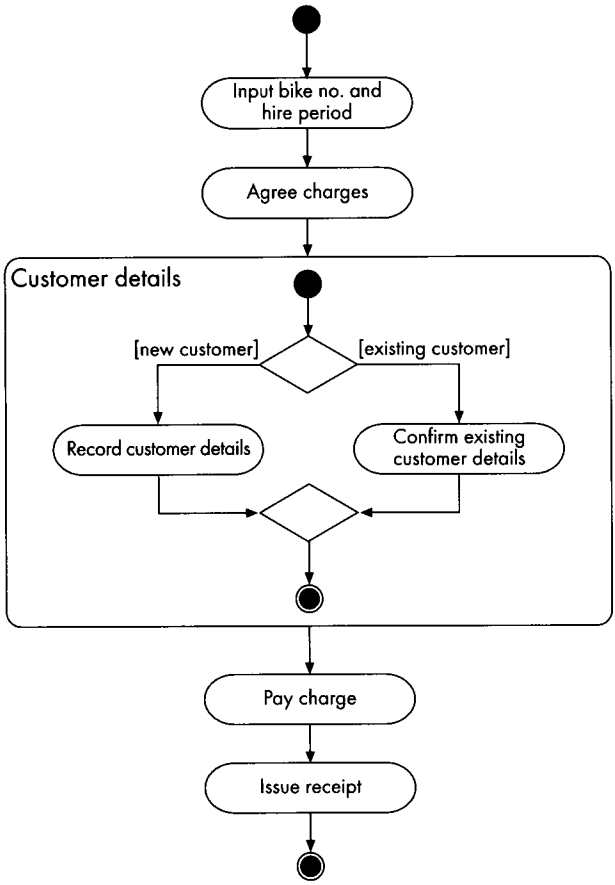


Figure 8.13 *Activity diagram of the 'Issue bike' use case, showing the subset of activities that relate to handling customer details*

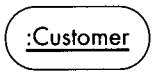


Figure 8.14 *Activities may be given the name of an associated object*

is drawn from the object to the activity. If an activity creates or updates an object, an object flow is drawn from the activity to the object.

Figure 8.15 is the activity diagram of the 'Issue bike' use case (see also Figure 8.5) including the objects that are involved in the use case and the object flows that link them to specific activities.

In the case where the state of an object is altered by an activity, this can be shown as in the label on the object in Figure 8.16, which shows how the activity 'Update customer record' updates the relevant customer object.

Where the object flows imply a transition between objects, the transitions themselves can be omitted. This can be seen in Figure 8.17a and b, which shows part of the 'Issue bike' use case.

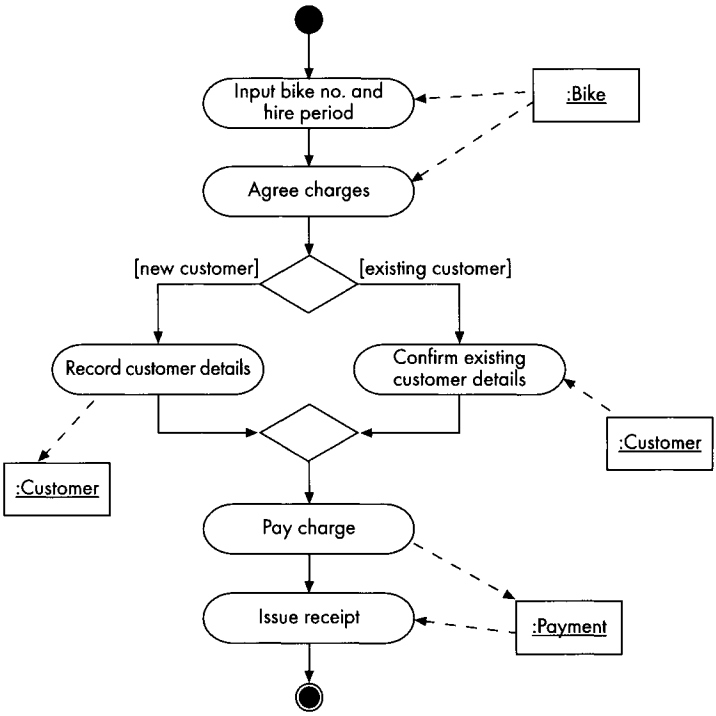


Figure 8.15 *Activity diagram for the 'Issue bike' use case, including associated objects and object flows*

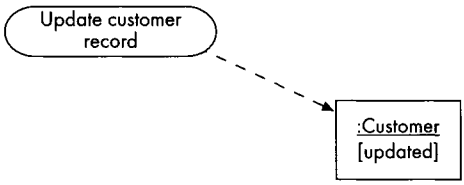


Figure 8.16 *Activity 'Update customer record' updates the relevant Customer object*

Figure 8.17a includes the transition between the activities 'Pay charge' and 'Issue receipt'; in Figure 8.17b these are replaced by the Payment object and the object flows linking the object to the activities.

Common problems

1 How do I know what makes a useful activity?

An activity is a situation in which something is happening. To identify useful activities, you need to mentally step through the process, use case or operation that you are describing and work out what has to be done and in what order.

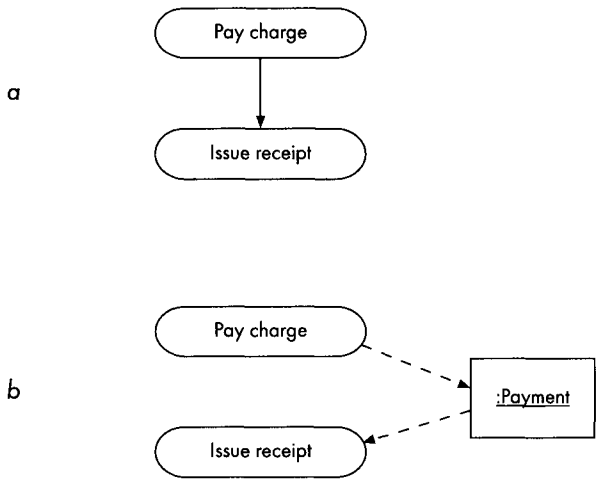


Figure 8.17 *a Part of the activity diagram for the 'Issue bike' use case, showing transition between two activities*
b Part of the activity diagram for the 'Issue bike' use case, replacing the transition with the Payment object and object flows between it and the two activities

It is useful to remember to name all activities with an active verb, such as 'Record customer details' or 'Calculate remaining deposit'. It is important not to confuse an activity, for example 'Archive hire details', with the state, 'Archived', that a Hire object can be in.

You need to be clear about the level of the activity diagram that you are drawing and what exactly it aims to describe, whether that is a high-level business workflow, a use case, or the details of an operation. An activity such as 'Add cost of hire to deposit' is appropriate as part of the activity diagram for the 'Get charges operation' (see Figure 8.2), but would be too detailed for a diagram describing the 'Issue bike' use case.

2 How do I know whether to include swimlanes, object flows or subsections in a diagram?

We have discussed all these techniques in the chapter because each of them offers a way of adding useful information to the basic activity diagram. The key word, however, is 'useful'. In the initial stages of development, for example, nobody has begun to think in detail about the objects in the system and there is no point in worrying about objects or object flows. In the same way, partitioning a diagram into subsets of activities can come in handy when the diagram is large and complex, but it should only be done to make the overall diagram easier to read not just to look clever. Techniques and diagrams are tools that

are there to help the developer, not to dictate what should or should not be included in a diagram.

3 How do I model activities that are a mixture of human actor action and computer action like calculating the remaining deposit?

It is quite straightforward to specify that some actions are manual (e.g. 'Check bike') and some are done using the system (e.g. 'Check return date'). This can be done with swimlanes, as in Figure 8.10 where the mechanic checks the bike and the computer checks the return date.

In the case where you want to show that one activity is a mixture of human and system actions, you would have to split the activity into two. In Figure 8.10 the activity of dealing with the return of a deposit is split into 'Calculate amount of deposit to return' (Computer) and 'Return deposit' (Receptionist).

Chapter summary

This chapter introduces activity diagrams, which are used at various stages of the development process to model the flow of actions and the decisions that cause them to take place. They can be used in the early stages of development to describe high level business workflows, then later to model use cases, and finally to clarify the details of individual operations on classes.

Activity diagrams can model sequencing, selection and repetition of activities. They can also show where activities may be carried out in parallel. It is possible to divide activity diagrams into swimlanes, indicating which person, organization or object has responsibility for which activities in the diagram. During detailed design it is also possible to indicate the input that an activity needs from a specific object, and how an object is affected by the output from an activity. However, activity diagrams are often most useful when used in their simplest form as a means for the developer and client to build a shared understanding of how the system works.

Bibliography

Bennett, S., McRobb, S. and Farmer, R. (2002) *Object-Oriented Systems Analysis and Design Using UML* (2nd edition), McGraw-Hill, London.

Fowler, M. (2000) *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (2nd edition), Addison-Wesley, Reading, MA.

There are a large number of references on the World Wide Web, including:
http://www.modelingstyle.info/activityDiagram.html

http://sunset.usc.edu/classes/cs577a_2000/papers/ActivitydiagramsforRoseArchitect.pdf

Quick check questions

You can find the answers to these in the chapter.

a In what way do activities differ from states as found in state diagrams?

b What types of process can activity diagrams model during the course of development?

c What common features of most programming languages can be modelled by an activity diagram?

d If an activity diagram shows two or more activities carried out in parallel does this mean that the activities must be carried out at the same time or that the order of processing does not matter?

e What does the diamond symbol mean on an activity diagram?

f What do two synchronization bars indicate in an activity diagram?

g What do swimlanes show in an activity diagram?

Exercises

8.1 Figure 8.18 shows a simple sequential activity diagram for the purchase of tickets by telephone. Modify the diagram to show that the activities 'Calculate total cost' and 'Record customer details' can be carried out in any order.

8.2 The owners of a small retail company make regular orders to their supplier. First, they check their current stock, and then compile the order. When they receive the goods, they check them against the order, and update stock levels. They also pay the supplier's bill.

Draw an activity diagram to illustrate the ordering process. Your diagram should include an example of parallel processing.

8.3 Figure 8.19 shows a simple sequential activity diagram illustrating a car entering a car park. Add swimlanes to the diagram to show which of the activities are carried out by

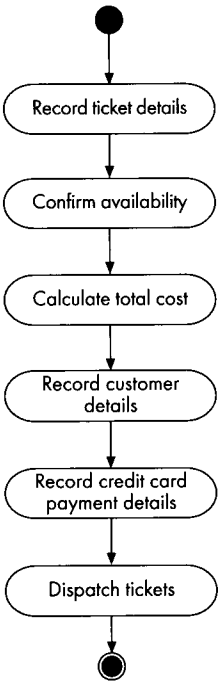


Figure 8.18 *Simple activity diagram for the purchase of tickets by telephone*

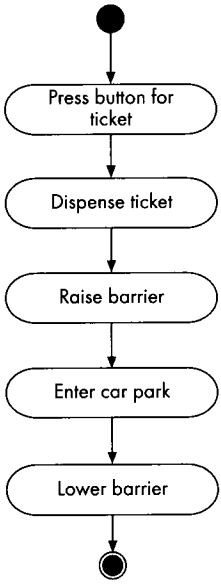


Figure 8.19 *Simple activity diagram illustrating a car entering a car park*

the driver of the car and which by the machine that controls entry to the car park.

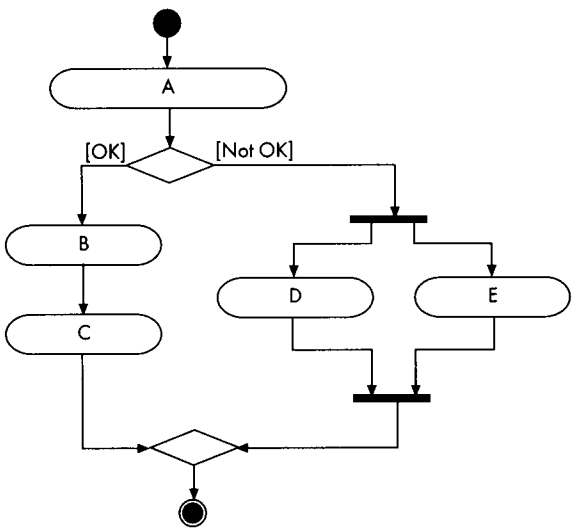
8.4 When a driver wants to leave the car park, he or she drives to the exit barrier and puts the ticket into the machine, which

calculates the amount owing. The driver inserts the money and if it is more than the amount owing, the machine gives change. The machine then raises the barrier, the driver drives out of the car park, and the machine lowers the barrier.

a Draw an activity diagram without swimlanes to illustrate what happens when a driver leaves the car park. Your diagram should include an example of alternative behaviours.

b Add swimlanes to your diagram to show who or what is responsible for the various activities.

8.5 Add swimlanes to Figure 8.5, the activity diagram for the 'Issue bike' use case, to show which activities are carried out by the receptionist, which by the computer and which by the customer.



a
```
do A
If OK then
    do B
    do C
Else
    do D or E (*not both*)
End
```

b
```
do A
If OK then
    do B
    do C
Else
do D and E (*in any order*)
End
```

c
```
do A
    do B
    do C
If Not OK then
    do D and E
End
```

d
```
do A
If Ok then
    do B
    do C
Else
    do D
    do E
End
```

Figure 8.20 Activity diagram and pseudocode for Exercise 8.6

8.6 Figure 8.20 shows an activity diagram followed by four extracts of pseudocode. Which one of the extracts a–d corresponds to the diagram?

8.7 Figure 8.21 is an activity diagram illustrating the purchase of a book over the Internet. Study the diagram and the statements that follow it. Indicate whether each statement is true or false according to the diagram.

a Customer details must be recorded before the amount owing is calculated.

b If the title is not available, a new title is input.

c The transaction must always be confirmed immediately after the credit card details are recorded.

d The credit card details must be recorded before the transaction is confirmed.

e Customer details are only recorded if the title chosen is available.

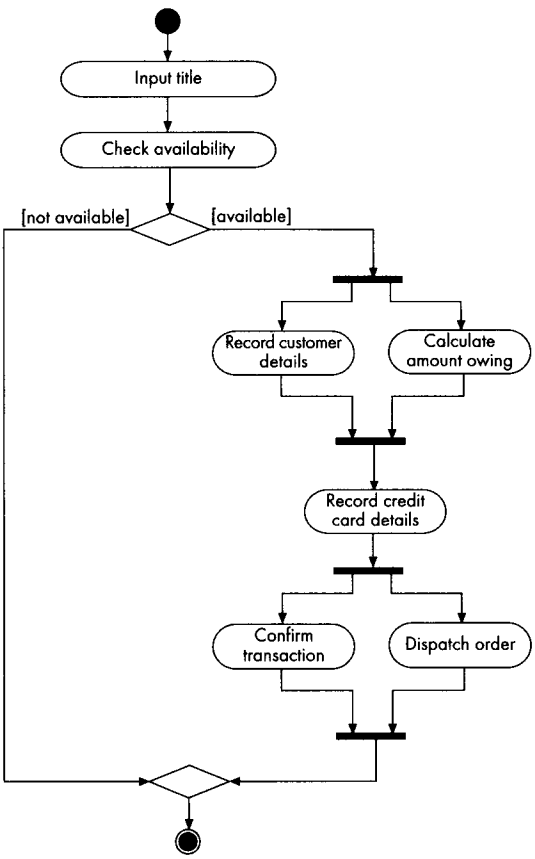
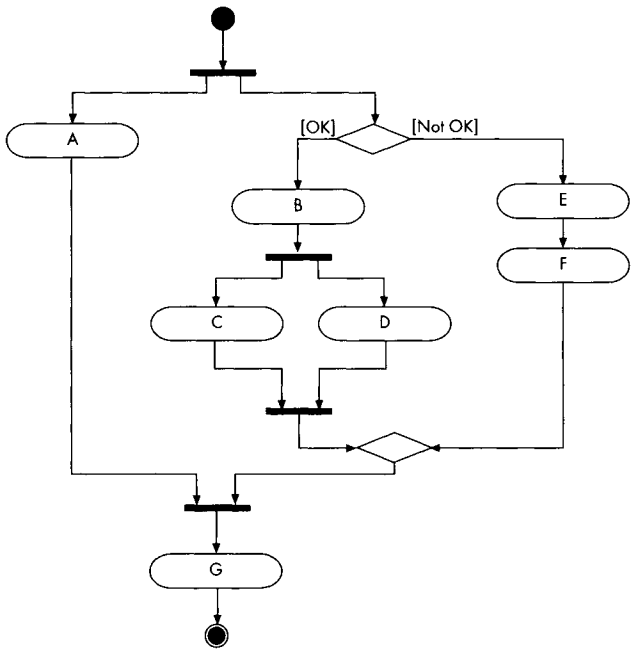


Figure 8.21 Activity diagram for the purchase of a book over the Internet



a
```
do A
If OK then
    do B
    do C and D (*in any order*)
do G
Else do E
     do F
do G
End
```

b
```
do A or
If OK then
    do B
    do C or D (*not both*)
Else do E
     do F
do G
End
```

c
```
do A and
If OK then
    do B
    do C
    do D
Else do E
     do F
do G
End
```

d
```
do A and
If OK then
    do B
    do C and D (*in any order*)
Else do E
     do F
do G
End
```

Figure 8.22 *Activity diagram and pseudocode for Exercise 8.8*

8.8 Figure 8.22 shows an activity diagram followed by four extracts of pseudocode. Which one of the extracts a–d corresponds to the diagram?

8.9 Modify the 'Issue bike' diagram in Figure 8.5 to show what happens when a customer hires more than one bike.

8.10 Modify the 'Handle bike return' diagram in Figure 8.9 to show what happens when a customer returns more than one bike.