# 6 Activity Planning

**Learning Objectives**

- Produce an activity plan for a project
- Estimate the overall duration of a project
- Create a critical path and a precedence network for a project

## 6.1 Introduction

In earlier chapters we looked at methods for forecasting the effort required for a project – both for the project as a whole and for individual activities. A detailed plan for the project, however, must also include a schedule indicating the start and completion times for each activity. This will enable us to:

- ensure that the appropriate resources will be available precisely when required;
- avoid different activities competing for the same resources at the same time;
- produce a detailed schedule showing which staff carry out each activity;
- produce a detailed plan against which actual achievement may be measured;
- produce a timed cash flow forecast;
- replan the project during its life to correct drift from the target.

To be effective, a plan must be stated as a set of targets, the achievement or non-achievement of which can be unambiguously measured. The activity plan does this by providing a target start and completion date for each activity (or a window within which each activity may be carried out). The starts and completions of activities must be clearly visible and this is one of the reasons why it is advisable to ensure that each and every project activity produces some tangible product or 'deliverable'. Monitoring the project's progress is then, at least in part, a case of ensuring that the products of each activity are delivered on time.

Project monitoring is discussed in more detail in Chapter 9.

As a project progresses it is unlikely that everything will go according to plan. Much of the job of project management concerns recognizing when something has gone wrong, identifying its causes and revising the plan to mitigate its effects. The activity plan should provide a means of evaluating the consequences of not meeting any of the activity target dates and guidance as to how the plan might most effectively be modified to bring the project back to target. We shall see that the activity plan may well also offer guidance as to which components of a project should be most closely monitored.

## 6.2 Objectives of Activity Planning

In addition to providing project and resource schedules, activity planning aims to achieve a number of other objectives which may be summarized as follows.

- *Feasibility assessment* Is the project possible within required timescales and resource constraints? In Chapter 5 we looked at ways of estimating the effort for various project tasks. However, it is not until we have constructed a detailed plan that we can forecast a completion date with any reasonable knowledge of its achievability. The fact that a project may have been estimated as requiring two work-years' effort might not mean that it would be feasible to complete it within, say, three months were eight people to work on it – that will depend upon the availability of staff and the degree to which activities may be undertaken in parallel.
- *Resource allocation* What are the most effective ways of allocating resources to the project. When should the resources be available? The project plan allows us to investigate the relationship between timescales and resource availability (in general, allocating additional resources to a project shortens its duration) and the efficacy of additional spending on resource procurement.
- *Detailed costing* How much will the project cost and when is that expenditure likely to take place? After producing an activity plan and allocating specific resources, we can obtain more detailed estimates of costs and their timing.
- *Motivation* Providing targets and being seen to monitor achievement against targets is an effective way of motivating staff, particularly where they have been involved in setting those targets in the first place.

Chapter 11 discusses motivation in more detail.

- *Coordination* When do the staff in different departments need to be available to work on a particular project and when do staff need to be transferred between projects? The project plan, particularly with large projects involving more than a single project team, provides an effective vehicle for communication and coordination among teams. In situations where staff may need to be transferred between project teams (or work concurrently on more than one project), a set of integrated project schedules should ensure that such staff are available when required and do not suffer periods of enforced idleness.

This coordination will normally form part of Programme Management.

Activity planning and scheduling techniques place an emphasis on completing the project in a minimum time at an acceptable cost or, alternatively, meeting a set target date at minimum cost. These are not, in themselves, concerned with meeting quality targets, which generally impose constraints on the scheduling process.

One effective way of shortening project durations is to carry out activities in parallel. Clearly we cannot undertake all the activities at the same time – some require the completion of others before they can start and there are likely to be resource constraints limiting how much may be done simultaneously. Activity scheduling will, however, give us an indication of the cost of these constraints in terms of lengthening timescales and provide us with an indication of how timescales may be shortened by relaxing those constraints. If we

try relaxing precedence constraints by, for example, allowing a program coding task to commence before the design has been completed, it is up to us to ensure that we are clear about the potential effects on product quality.

## 6.3 When to Plan

Planning is an ongoing process of refinement, each iteration becoming more detailed and more accurate than the last. Over successive iterations, the emphasis and purpose of planning will shift.

During the feasibility study and project start-up, the main purpose of planning will be to estimate timescales and the risks of not achieving target completion dates or keeping within budget. As the project proceeds beyond the feasibility study, the emphasis will be placed upon the production of activity plans for ensuring resource availability and cash flow control.

Throughout the project, until the final deliverable has reached the customer, monitoring and replanning must continue to correct any drift that might prevent meeting time or cost targets.

## 6.4 Project Schedules

Before work commences on a project or, possibly, a stage of a larger project, the project plan must be developed to the level of showing dates when each activity should start and finish and when and how much of each resource will be required. Once the plan has been refined to this level of detail we call it a project schedule. Creating a project schedule comprises four main stages.

On a large project, detailed plans for the later stages will be delayed until information about the work required has emerged from the earlier stages.

The first step in producing the plan is to decide what activities need to be carried out and in what order they are to be done. From this we can construct an *ideal activity plan* – that is, a plan of when each activity would ideally be undertaken were resources not a constraint. It is the creation of the ideal activity plan that we shall discuss in this chapter. This activity plan is generated by Steps 4 and 5 of Step Wise (Figure 6.1).

The ideal activity plan will then be the subject of an activity risk analysis, aimed at identifying potential problems. This might suggest alterations to the ideal activity plan and will almost certainly have implications for resource allocation. Activity risk analysis is the subject of Chapter 7.

The third step is *resource allocation*. The expected availability of resources might place constraints on when certain activities can be carried out, and our ideal plan might need to be adapted to take account of this. Resource allocation is covered in Chapter 8.

The final step is *schedule production*. Once resources have been allocated to each activity, we will be in a position to draw up and publish a project schedule, which indicates planned start and completion dates and a resource requirements statement for each activity. Chapter 9 discusses how this is done and the role of the schedule in managing a project.

## 6.5 Projects and Activities

### Defining activities

Before we try to identify the activities that make up a project it is worth reviewing what we mean by a project and its activities and adding some assumptions that will be relevant when we start to produce an activity plan.

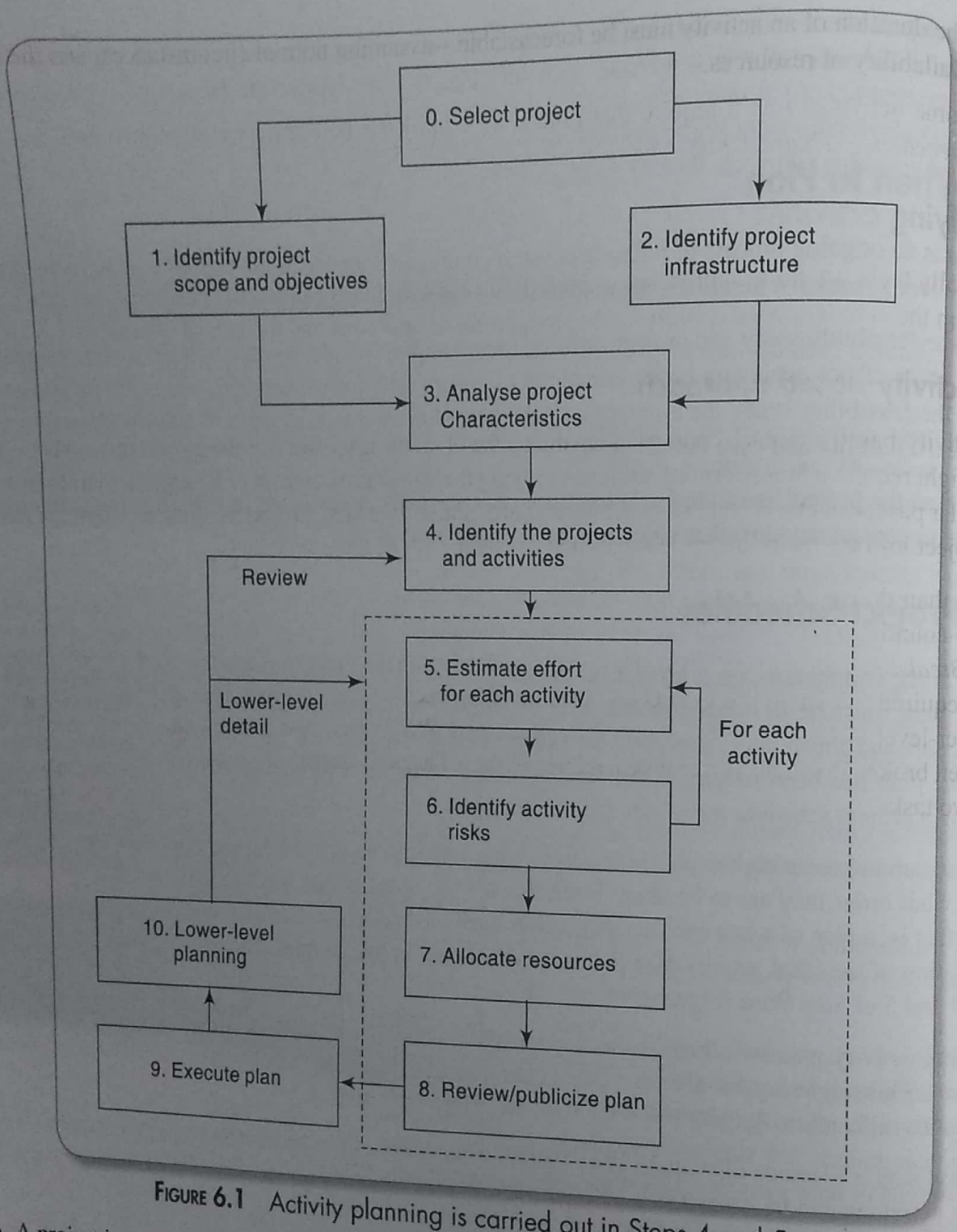



**FIGURE 6.1** Activity planning is carried out in Steps 4 and 5

- A project is composed of a number of interrelated activities.
- A project may start when at least one of its activities is ready to start.
- A project will be completed when all of the activities it encompasses have been completed.
- An activity must have a clearly defined start and a clearly defined end-point, normally marked by the production of a tangible deliverable.
- If an activity requires a resource (as most do) then that resource requirement must be forecastable and is assumed to be required at a constant level throughout the duration of the activity.

Activities must be defined so that they meet these criteria. Any activity that does not meet these criteria must be redefined.

- The duration of an activity must be forecastable – assuming normal circumstances, and the reasonable availability of resources.
- Some activities might require that others are completed before they can begin (these are known as *precedence requirements*).

## Identifying activities

Essentially there are three approaches to identifying the activities or tasks that make up a project – we shall call them the *activity-based approach, the product-based approach* and the *hybrid approach.*

## The activity-based approach

The activity-based approach consists of creating a list of all the activities that the project is thought to involve. This might require a brainstorming session involving the whole project team or it might stem from an analysis of similar past projects. When listing activities, particularly for a large project, it might be helpful to subdivide the project into the main life-cycle stages and consider each of these separately.

Rather than doing this in an ad hoc manner, with the obvious risks of omitting or double-counting tasks, a much favoured way of generating a task list is to create a *Work Breakdown Structure* (WBS). This involves identifying the main (or high-level) tasks required to complete a project and then breaking each of these down into a set of lower-level tasks. Figure 6.2 shows a fragment of a WBS where the design task has been broken down into three tasks and one of these has been further decomposed into two tasks.

WBSs are advocated by BS 6079, the British Standards Institution's *Guide to Project Management.*

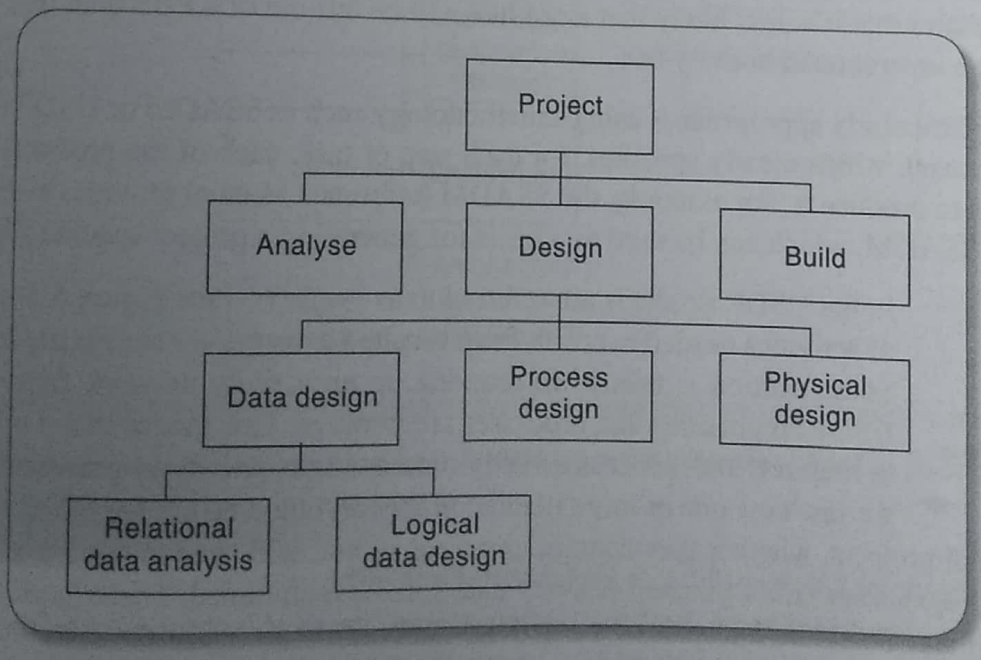



**Figure 6.2** A fragment of an activity-based Work Breakdown Structure

Activities are added to a branch in the structure if they contribute directly to the task immediately above – if they do not contribute to the parent task, then they should not be added to that branch. The tasks at each level in any branch should include everything that is required to complete the task at the higher level.

When preparing a WBS, consideration must be given to the final level of detail or depth of the structure. Too great a depth will result in a large number of small tasks that will be difficult to manage, whereas a too shallow structure will provide insufficient detail for project control. Each branch should, however, be broken down at least to a level where each leaf may be assigned to an individual or responsible section within the organization.

A complete task catalogue will normally include task definitions along with task input and output products and other task-related information.

Advantages claimed for the WBS approach include the belief that it is much more likely to result in a task catalogue that is complete and is composed of non-overlapping activities. Note that it is only the leaves of the structure that comprise the list of activities in the project – higher-level nodes merely represent collections of activities.

The WBS also represents a structure that may be refined as the project proceeds. In the early part of a project we might use a relatively high-level or shallow WBS, which can be developed as information becomes available, typically during the project's analysis and specification phases.

Once the project's activities have been identified (whether or not by using a WBS), they need to be sequenced in the sense of deciding which activities need to be completed before others can start.

## Product-based approach

The product-based approach, used in PRINCE2 and Step Wise, has already been described in Chapter 3. It consists of producing a Product Breakdown Structure and a Product Flow Diagram. The PFD indicates, for each product, which other products are required as inputs. The PFD can therefore be easily transformed into an ordered list of activities by identifying the transformations that turn some products into others. Proponents of this approach claim that it is less likely that a product will be left out of a PBS than that an activity might be omitted from an unstructured activity list.

This approach is particularly appropriate if using a methodology such as SSADM or USDP (Unified Software Development Process), which clearly specifies, for each step or task, each of the products required and the activities required to produce it. For example, the SSADM Reference Manual provides a set of generic PBSs for each stage in SSADM, which can be used as a basis for generating a project specific PBS.

See I. Jacobson, G. Booch and J. Rumbaugh (1999) *The Unified Software Development Process*, Addison-Wesley.

In the USDP, products are referred to as *artifacts* – see Figure 6.3 – and the sequence of activities needed to create them is called a *workflow*– see Figure 6.4 for an example. Some caution is needed in drawing up an activity network from these workflows. USDP emphasizes that processes are iterative. This means that it may not be possible to map a USDP process directly onto a single activity in a network. In Section 4.18 we saw how one or more iterated processes could be hidden in the single execution of a larger activity. All projects, whether they contain iterations or not, will need to have some fixed milestones or time-boxes if progress towards a planned delivery date is to be maintained. These larger activities with the fixed completion dates would be the basis of the activity network.

## Hybrid approach

BS 6079 states that WBSs may be product-based, cost-centre-based, task-based or function-based but that product-based WBSs are preferred.

The WBS illustrated in Figure 6.2 is based entirely on a structuring of activities. Alternatively, and perhaps more commonly, a WBS may be based upon the project's products as illustrated in Figure 6.5, which is in turn based on a simple list of final deliverables and, for each deliverable, a set of activities required to produce that product. Figure 6.5 illustrates a flat WBS and it is likely that, in a project of any size, it would be beneficial to introduce additional levels – structuring both products and

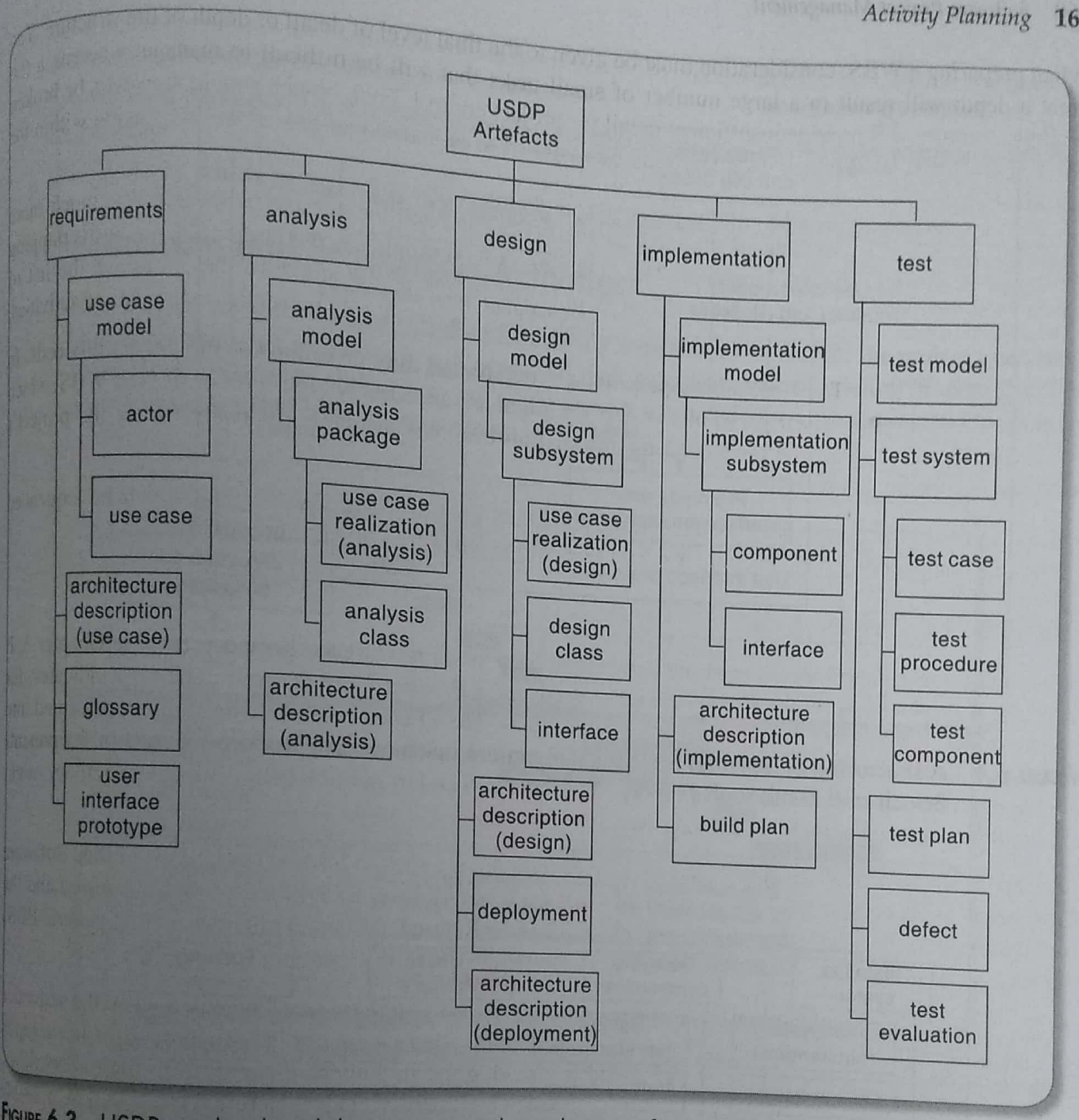



**FIGURE 6.3** USDP product breakdown structure based on artefacts identified in Jacobson, Booch and Rumbaugh (1999)

activities. The degree to which the structuring is product-based or activity-based might be influenced by the nature of the project and the particular development method adopted. As with a purely activity-based WBS, having identified the activities we are then left with the task of sequencing them.

A framework dictating the number of levels and the nature of each level in the structure may be imposed on a WBS. For example, in their MITP methodology, IBM recommend that the following five levels should be used in a WBS:

- *Level 1: Project*
- *Level 2: Deliverables* such as software, manuals and training courses

Not all of the products in this activity structuring will be final products. Some will be further refined in subsequent steps.

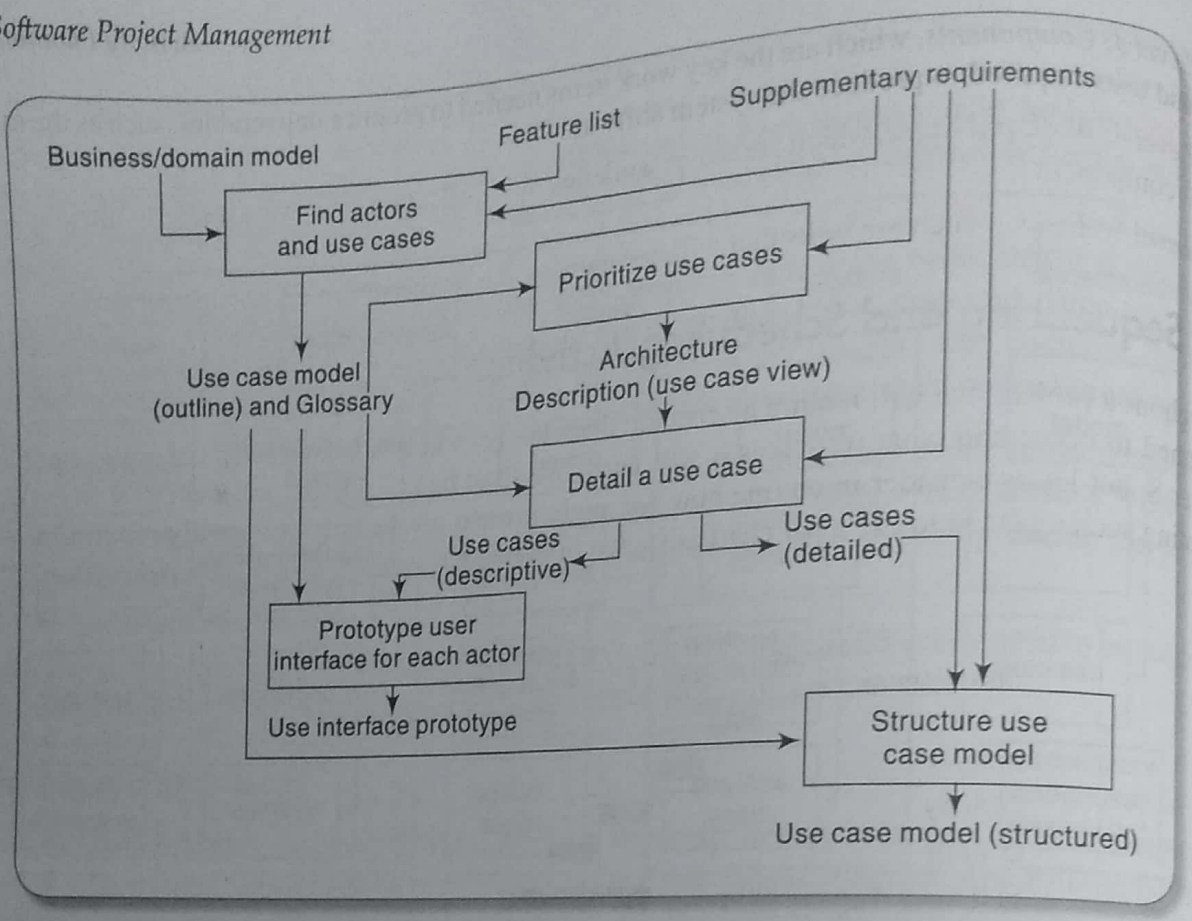



**FIGURE 6.4** A structuring of activities for the USDP requirements capture workflow based on Jacobson, Booch and Rumbaugh (1999)

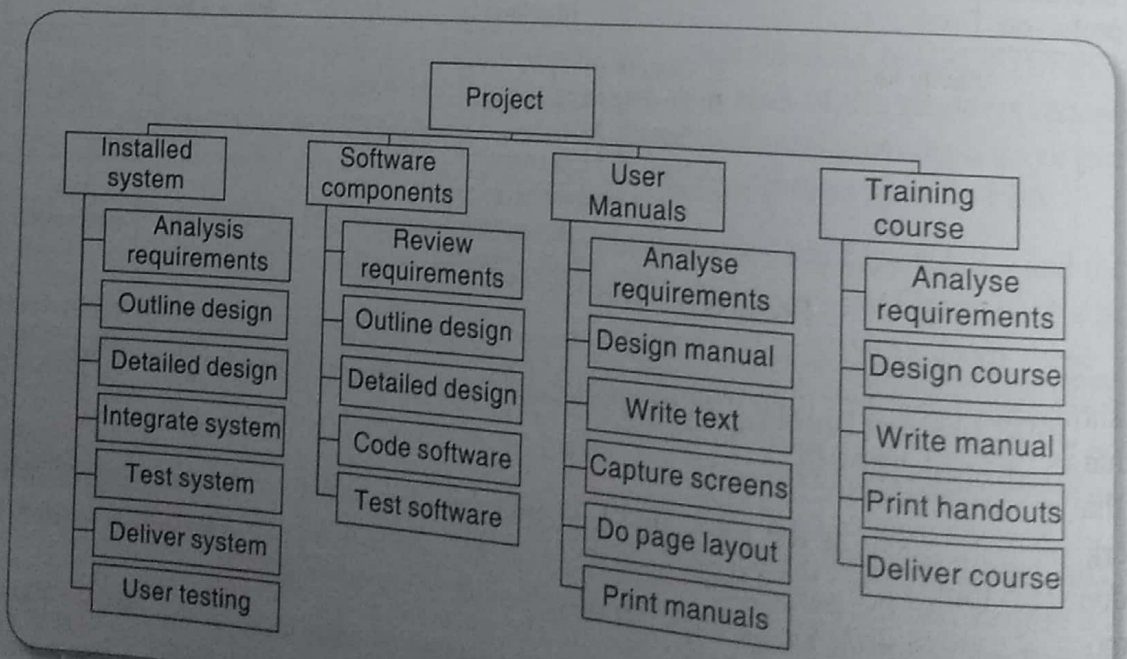



**FIGURE 6.5** A hybrid Work Breakdown Structure based on deliverables and activities

- *Level 3: Components*, which are the key work items needed to produce deliverables, such as the modules and tests required to produce the system software
- *Level 4: Work-packages*, which are major work items, or collections of related tasks, required to produce a component
- *Level 5: Tasks*, which are tasks that will normally be the responsibility of a single person

## 6.6 Sequencing and Scheduling Activities

Throughout a project, we will require a schedule that clearly indicates when each of the project's activities is planned to occur and what resources it will need. We shall be considering scheduling in more detail in Chapter 8, but let us consider in outline how we might present a schedule for a small project. One way of presenting such a plan is to use a bar chart as shown in Figure 6.6.

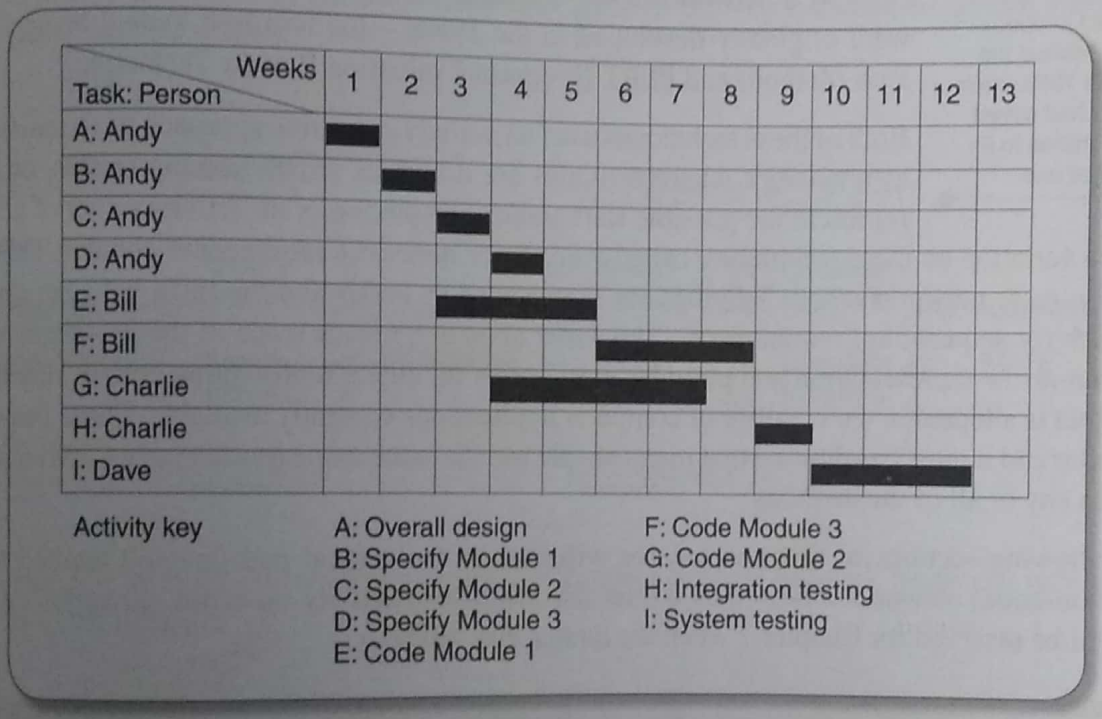



**FIGURE 6.6** A project plan as a bar chart

The chart shown has been drawn up taking account of the nature of the development process (that is, certain tasks must be completed before others may start) and the resources that are available (for example, activity C follows activity B because Andy cannot work on both tasks at the same time). In drawing up the chart, we have therefore done two things – we have sequenced the tasks (that is, identified the dependencies among activities dictated by the development process) and scheduled them (that is, specified when they should take place). The scheduling has had to take account of the availability of staff and the ways in which the activities have been allocated to them. The schedule might look quite different were there a different number of staff or were we to allocate the activities differently.

The bar chart does not show why certain decisions have been made. It is not clear, for example, why activity H is not scheduled to start until week 9. It could be that it cannot start until activity F has been completed or it might be because Charlie is going to be on holiday during week 8.

Separating the logical sequencing from the scheduling may be likened to the principle in systems analysis of separating the logical system from its physical implementation.

In the case of small projects, this combined sequencing–scheduling approach might be quite suitable, particularly where we wish to allocate individuals to particular tasks at an early planning stage. However, on larger projects it is better to separate out these two activities: to sequence the tasks according to their logical relationships and then to schedule them taking into account resources and other factors.

Approaches to scheduling that achieve this separation between the logical and the physical use networks to model the project and it is these approaches that we will consider in subsequent sections of this chapter.

## 6.7 Network Planning Models

CPM was developed by the DuPont Chemical Company which published the method in 1958, claiming that it had saved them $1 million in its first year of use.

These project scheduling techniques model the project's activities and their relationships as a network. In the network, time flows from left to right. These techniques were originally developed in the 1950s – the two best known being CPM (Critical Path Method) and PERT (Program Evaluation Review Technique).

Both of these techniques used an *activity-on-arrow* approach to visualizing the project as a network where activities are drawn as arrows joining circles, or nodes, which represent the possible start and/or completion of an activity or set of activities. More recently a variation on these techniques, called *precedence networks*, has become popular. This method uses *activity-on-node* networks where activities are represented as nodes and the links between nodes represent precedence (or sequencing) requirements. This latter approach avoids some of the problems inherent in the activity-on-arrow representation and provides more scope for easily representing certain situations. It is this method that is adopted in the majority of computer applications currently available. These three methods are very similar and it must be admitted that many people use the same name (particularly CPM) indiscriminately to refer to any or all of the methods.

In the following sections of this chapter, we will look at the critical path method applied to precedence (activity-on-node) networks followed by a brief introduction to activity-on-arrow networks – a discussion of PERT will be reserved for Chapter 7 when we look at risk analysis.

**Case Study** Examples

In Chapter 2 we saw how Amanda identified that three new software components would need to be developed and a further component would need to be rewritten. Figure 6.7 shows the fragment of a network that she has developed as an activity-on-node network. Figure 6.8 shows how this network would look represented as an activity-on-arrow network. Study each of the networks briefly to verify that they are, indeed, merely different graphical representations of the same thing.

## 6.8 Formulating a Network Model

The first stage in creating a network model is to represent the activities and their interrelationships as a graph. In activity-on-node we do this by representing activities as nodes (boxes) in the graph – the lines between nodes represent dependencies.

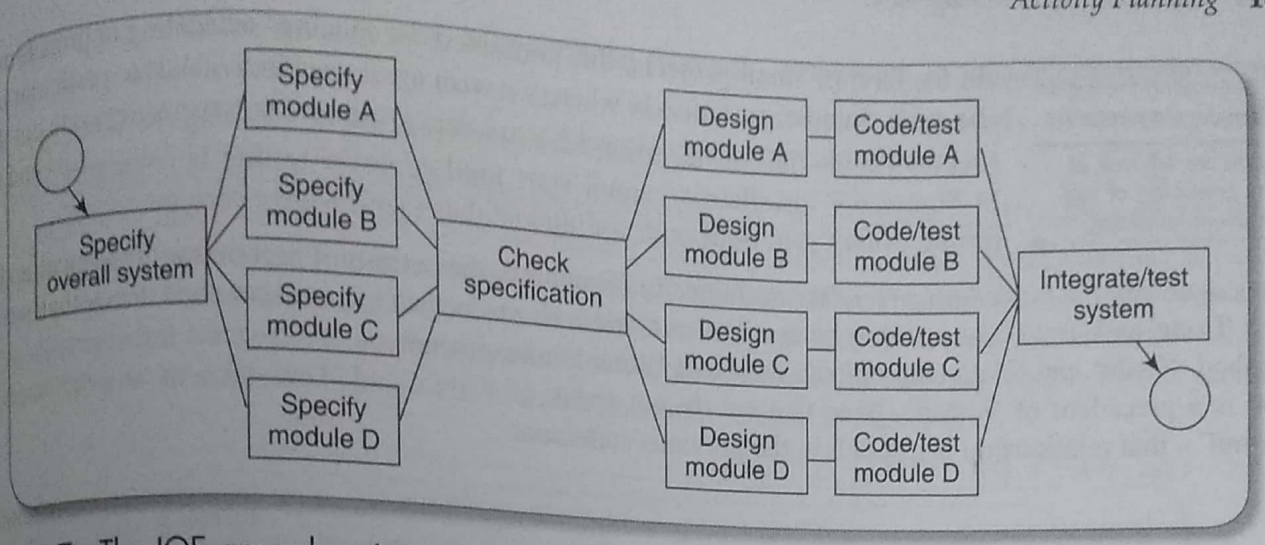



**FIGURE 6.7** The IOE annual maintenance contracts project activity network fragment with a checkpoint activity added

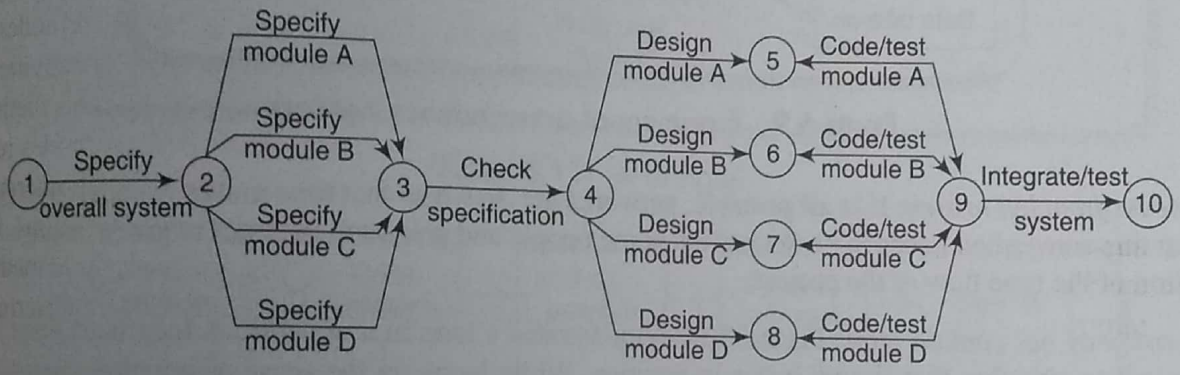



**FIGURE 6.8** The IOE annual maintenance contracts project activity network fragment represented as a CPM network

## Constructing precedence networks

Before we look at how networks are used, it is worth spending a few moments considering some rules for their construction.

*A project network should have only one start node* Although it is logically possible to draw a network with more than one starting node, it is undesirable to do so as it is a potential source of confusion. In such cases (for example, where more than one activity can start immediately the project starts) it is normal to invent a 'start' activity which has zero duration but may have an actual start date.

*A project network should have only one end node* The end node designates the completion of the project and a project may finish only once! Although it is possible to draw a network with more than one end node, it will almost certainly lead to confusion if this is done. Where the completion of a project depends upon more than one 'final' activity it is normal to invent a 'finish' activity.

*A node has duration* A node represents an activity and, in general, activities take time to execute. Notice, however, that the network in Figure 6.7 does not contain any reference to durations. This network drawing

merely represents the logic of the project – the rules governing the order in which activities are to be carrie out.

Later we will look at the possibility of 'lag' between activities.

*Links normally have no duration* Links represent the relationships between activitie In Figure 6.9 installation cannot start until program testing is complete. Progra testing cannot start until both coding and data take-on have been completed.

*Precedents are the immediate preceding activities* In Figure 6.9, the activity 'Program test' cannot start unti both 'Code' and 'Data take-on' have been completed and activity 'Install' cannot start until 'Program test' ha finished. 'Code' and 'Data take-on' can therefore be said to be precedents of 'Program test', and 'Progra test' is a precedent of 'Install'. Note that we do not speak of 'Code' and 'Data take-on' as precedents o 'Install' – that relationship is implicit in the previous statement.

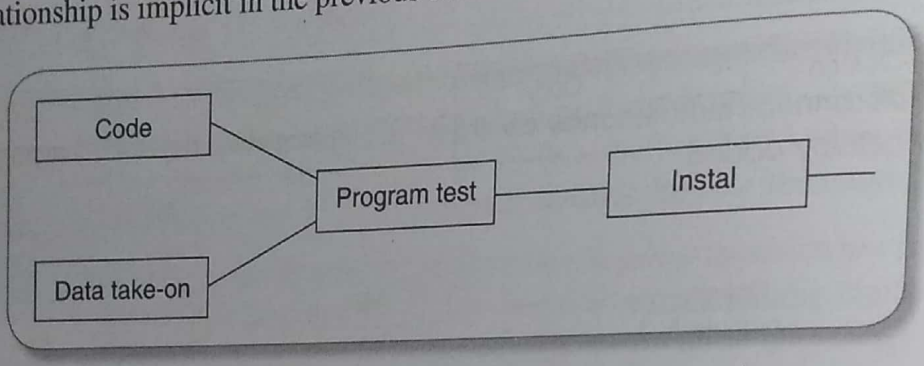



**FIGURE 6.9** Fragment of a precedence network

*Time moves from left to right* If at all possible, networks are drawn so that time moves from left to right. It is rare that this convention needs to be flouted but some people add arrows to the lines to give a stronger visual indication of the time flow of the project.

*A network may not contain loops* Figure 6.10 demonstrates a loop in a network. A loop is an error in that it represents a situation that cannot occur in practice. While loops, in the sense of iteration, may occur in practice, they cannot be directly represented in a project network. Note that the logic of Figure 6.10 suggests that program testing cannot start until the errors have been corrected.

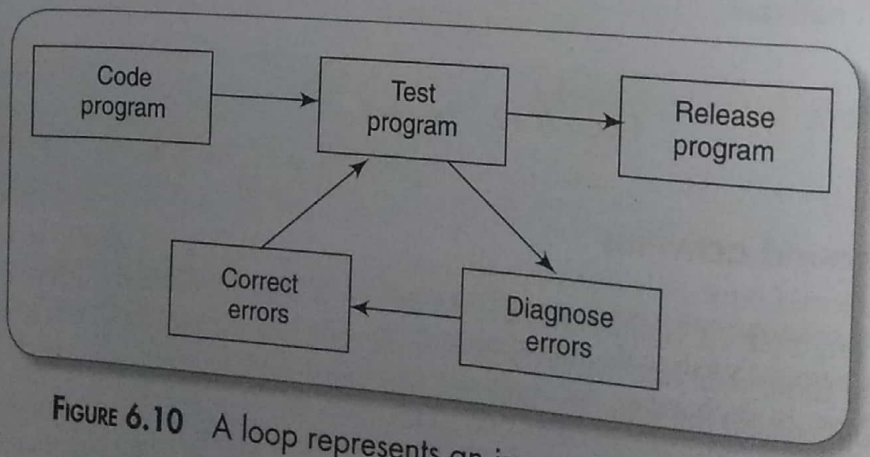



**FIGURE 6.10** A loop represents an impossible sequence

If we know the number of times we expect to repeat a set of activities, a test– diagnose–correct sequence, for example, then we can draw that set of activities as a straight sequence, repeating it the appropriate number of times. If we do not know how many times a sequence is going to be repeated then we cannot calculate the duration of the project unless we adopt an alternative strategy such as redefining the complete sequence as a single activity and estimating how long it will take to complete it.

Although it is easy to see the loop in this simple network fragment, very large networks can easily contain complex loops which are difficult to spot when they are initially constructed. Fortunately, all network planning applications will detect loops and generate error messages when they are found.

*A network should not contain dangles* A dangling activity such as 'Write user manual' in Figure 6.11 should not exist as it is likely to lead to errors in subsequent analysis. Indeed, in many cases dangling activities indicate errors in logic when activities are added as an afterthought. If, in Figure 6.11, we mean to indicate that the project is complete once the software has been installed *and* the user manual written then we should redraw the network with a final completion activity – which, at least in this case, is probably a more accurate representation of what should happen. The redrawn network is shown in Figure 6.12.

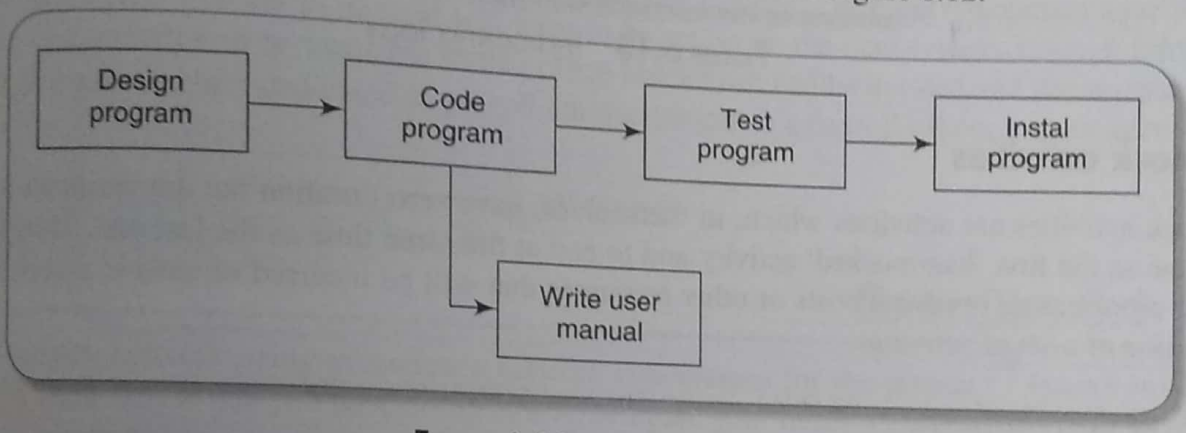



**FIGURE 6.11** A dangle

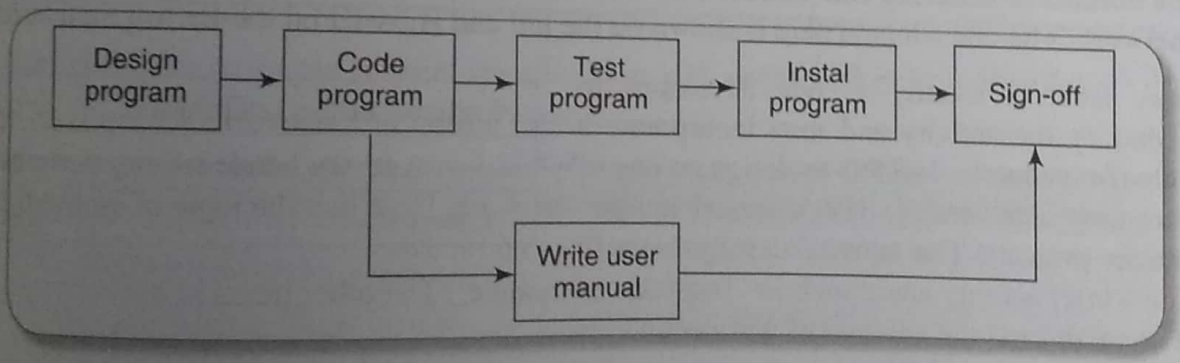



**FIGURE 6.12** Resolving the dangle

## Representing lagged activities

We might come across situations where we wish to undertake two activities in parallel so long as there is a lag between the two. We might wish to document amendments to a program as it is being tested – particularly if evaluating a prototype. In such a case we could designate an activity 'test and document amendments'. This would, however, make it impossible to show that amendment recording could start, say, one day after testing had begun and finish a little after the completion of testing.

Where activities can occur in parallel with a time lag between them, we represent the lag with a duration on the linking arrow as shown in Figure 6.13. This indicates that documenting amendments can start one day after the start of prototype testing and will be completed two days after prototype testing is completed.

Documenting amendments may take place alongside prototype testing so long as it starts at least one day later and finishes two days later.

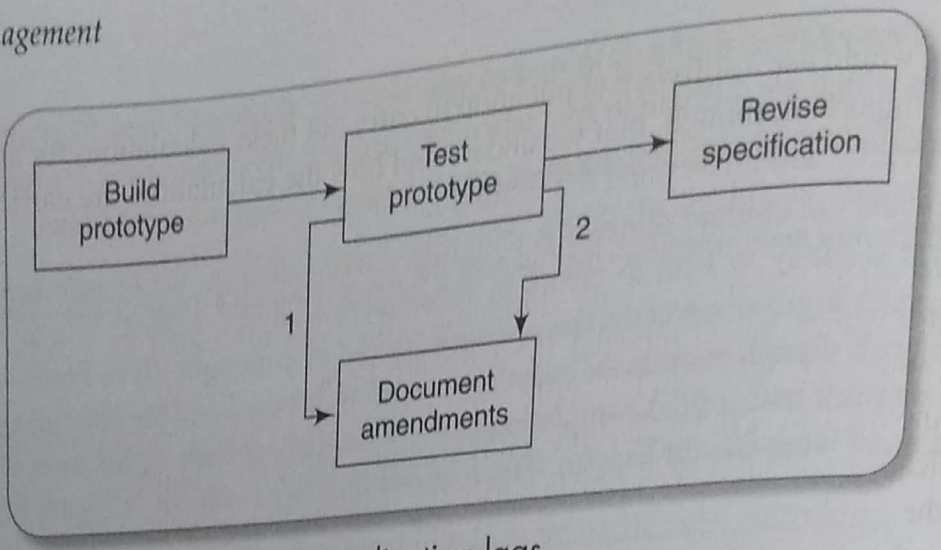



FIGURE 6.13 Indicating lags

### Hammock activities

Hammock activities are activities which, in themselves, have zero duration but are assumed to start at the same time as the first 'hammocked' activity and to end at the same time as the last one. They are normally used for representing overhead costs or other resources that will be incurred or used at a constant rate over the duration of a set of activities.

### Labelling conventions

There are a number of differing conventions that have been adopted for entering information on an activity-on-node network. The one adopted here is shown on the left and is based on the British Standard BS 4335.

The activity label is usually a code developed to uniquely identify the activity and may incorporate a project code (for example, IoE/P/3 to designate one of the programming activities for IOE's annual maintenance contract project). The activity description will normally be a brief activity name such as 'Test take-on module'. The other items in our activity node will be explained as we discuss the analysis of a project network.

| Earliest start | Duration | Earliest finish |
|---|---|---|
| Activity label, activity description | | |
| Latest start | Float | Latest finish |

## 6.9 Adding the Time Dimension

Having created the logical network model indicating what needs to be done and the interrelationships between those activities, we are now ready to start thinking about when each activity should be undertaken.

The critical path approach is concerned with two primary objectives: planning the project in such a way that it is completed as quickly as possible; and identifying those activities where a delay in their execution is likely to affect the overall end date of the project or later activities' start dates.

The method requires that for each activity we have an estimate of its duration. The network is then analysed by carrying out a *forward pass*, to calculate the earliest dates at which activities may commence and the project be completed, and a *backward pass*, to calculate the latest start dates for activities and the *critical path*.

In practice we would use a software application to carry out these calculations for anything but the smallest of projects. It is important, though, that we understand how the calculations are carried out in order to interpret the results correctly and understand the limitations of the method.

The description and example that follow use the small example project outlined in Table 6.1 – a project composed of eight activities whose durations have been estimated as shown in the table. Brigette at Brightmouth College has completed the software package evaluation and a software package has been chosen and approved. Now that the application software is known, the hardware needed as a platform can be acquired. Another task will be 'system configuration' – there are a number of parameters that will have to set in the application so that it runs satisfactorily for Brightmouth College. Once the parameters have been set, details of the employees who are to be paid will have to be set up on the new system. Enough information about the new system will now be available so that office procedures can be devised and documented. There are currently no staff currently dedicated to payroll administration so a payroll officer is to be recruited and then trained.

## Exercise 6.1

Draw an activity network using precedence network conventions for the project specified in Table 6.1. When you have completed it, compare your result with that shown in Figure 6.14.

Figure 6.14 illustrates the network for the project specified in Table 6.1.

**TABLE 6.1** An example project specification with estimated activity durations and precedence requirements

| Activity | Duration (weeks) | Precedents |
|---|---|---|
| A Hardware selection | 6 | |
| B System configuration | 4 | |
| C Install hardware | 3 | A |
| D Data migration | 4 | B |
| E Draft office procedures | 3 | B |
| F Recruit staff | 10 | |
| G User training | 3 | E, F |
| H Install and test system | 2 | C, D |

## 6.10 The Forward Pass

The forward pass is carried out to calculate the earliest dates on which each activity may be started and completed.

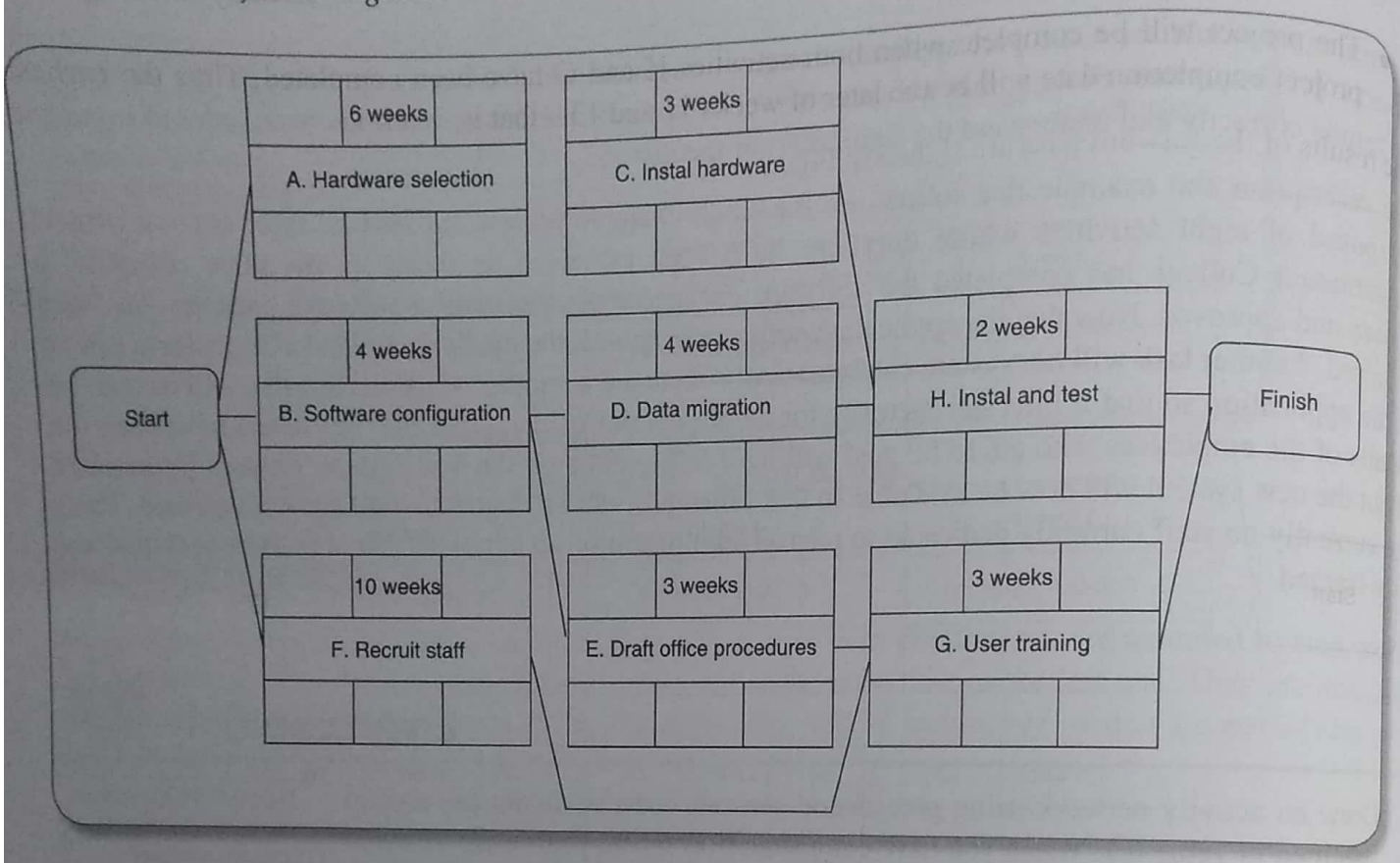



**FIGURE 6.14** The precedence network for the example project

Where an actual start date is known, the calculations may be carried out using actual dates. Alternatively we can use day or week numbers and that is the approach we shall adopt here. By convention, dates indicate the end of a period and the project is therefore shown as starting at the end of week zero (or the beginning of week 1).

During the forward pass, earliest dates are recorded as they are calculated.

The forward pass rule: the earliest start date for an activity is the earliest finish date for the preceding activity. Where there is more than one immediately preceding activity we take the *latest* of the *earliest finish dates* for those activities.

The forward pass and the calculation of earliest start dates are carried out according to the following reasoning.

- Activities A, B and F may start immediately, so the earliest date for their start is zero.
- Activity A will take 6 weeks, so the earliest it can finish is week 6.
- Activity B will take 4 weeks, so the earliest it can finish is week 4.
- Activity F will take 10 weeks, so the earliest it can finish is week 10.
- Activity C can start as soon as A has finished so its earliest start date is week 6. It will take 3 weeks so the earliest it can finish is week 9.
- Activities D and E can start as soon as B is complete so the earliest they can each start is week 4. Activity D, which will take 4 weeks, can therefore finish by week 8 and activity E, which will take 3 weeks, can therefore finish by week 7.
- Activity G cannot start until both E and F have been completed. It cannot therefore start until week 10 – the later of weeks 7 (for activity E) and 10 (for activity F). It takes 3 weeks and finishes in week 13.
- Similarly, Activity H cannot start until week 9 – the later of the two earliest finish dates for the preceding activities C and D.

- The project will be complete when both activities H and G have been completed. Thus the earliest project completion date will be the later of weeks 11 and 13 – that is, week 13.

The results of the forward pass are shown in Figure 6.15.

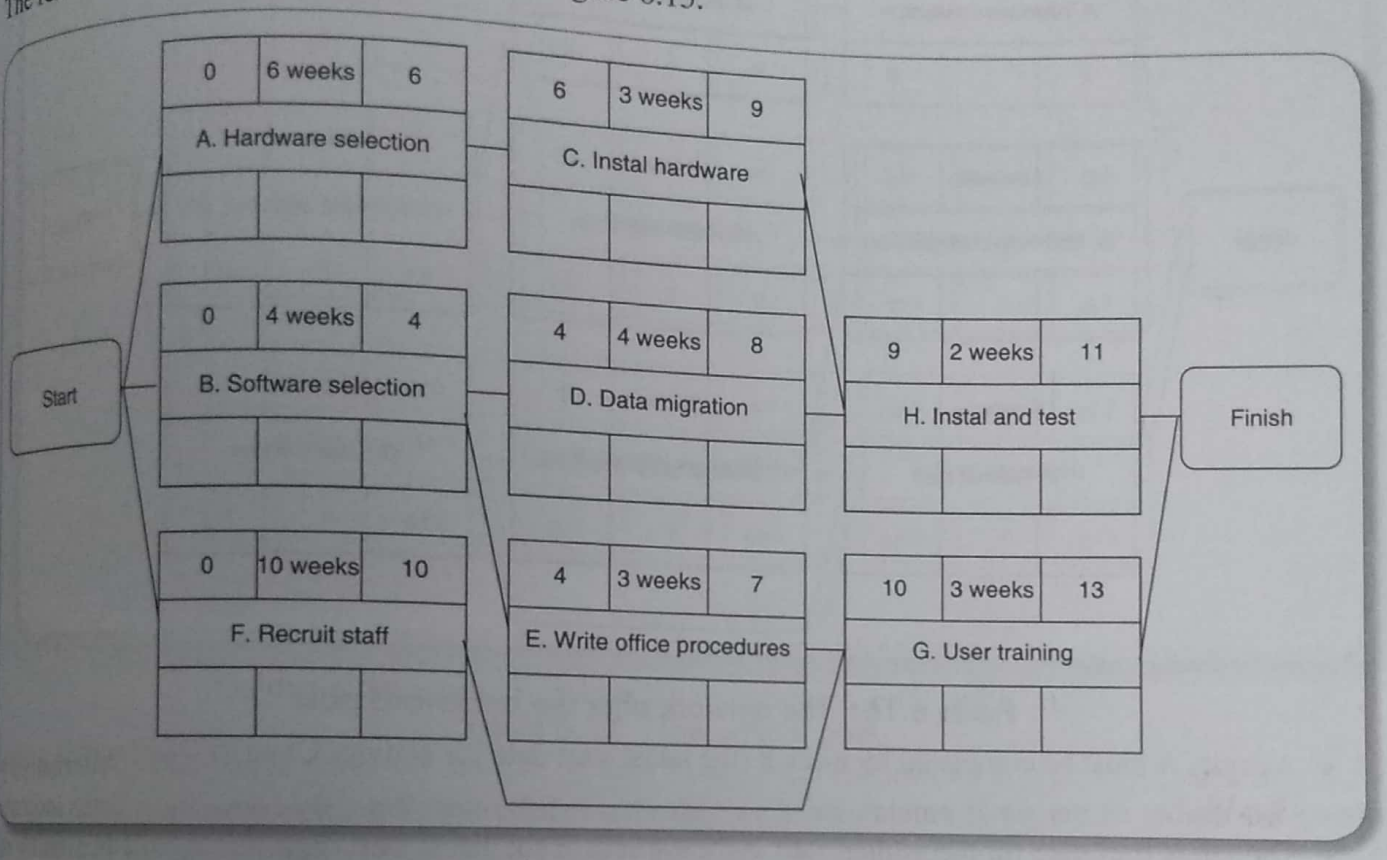



**FIGURE 6.15** The network after the forward pass

## 6.11 Backward Pass

The second stage in the analysis of a critical path network is to carry out a backward pass to calculate the latest date at which each activity may be started and finished without delaying the end date of the project. In calculating the latest dates, we assume that the latest finish date for the project is the same as the earliest finish date – that is, we wish to complete the project as early as possible.

The backward pass rule: the latest finish date for an activity is the latest start date for the activity that commences immediately that activity is complete. Where more than one activity can commence we take the earliest of the latest start dates for those activities.

Figure 6.16 illustrates our network after carrying out the backward pass. The latest activity dates are calculated as follows.

- The latest completion date for activities G and H is assumed to be week 13.
- Activity H must therefore start at week 11 at the latest (13 – 2) and the latest start date for activity G is week 10 (13 – 3).
- The latest completion date for activities C and D is the latest date at which activity H must start – that is, week 11. They therefore have latest start dates of week 8 (11 – 3) and week 7 (11 – 4) respectively.
- Activities E and F must be completed by week 10 so their earliest start dates are weeks 7 (10 – 3) and 0 (10 – 10) respectively.
- Activity B must be completed by week 7 (the latest start date for both activities D and E) so its latest start is week 3 (7 – 4).

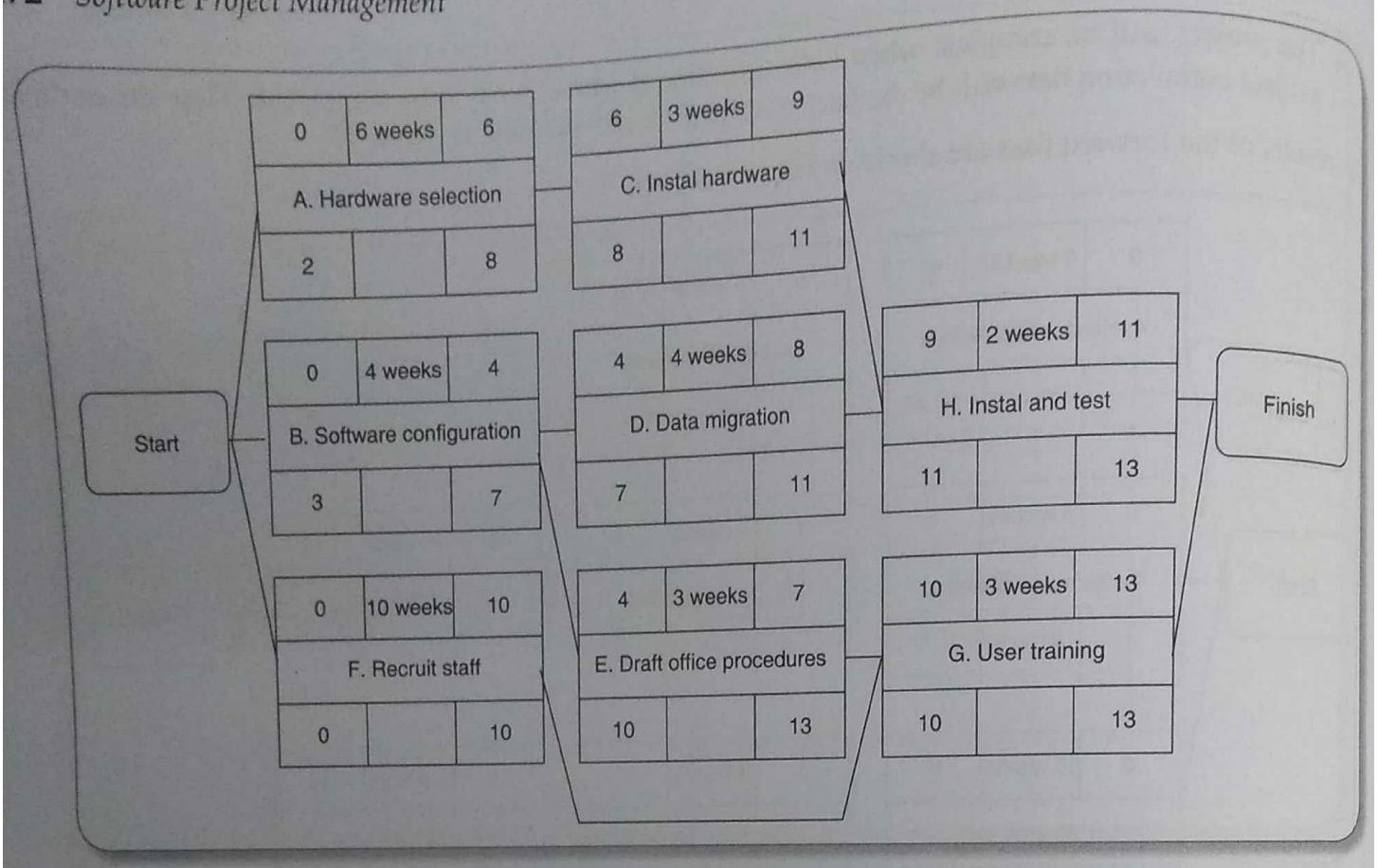



**FIGURE 6.16** The network after the backward pass

- Activity A must be completed by week 8 (the latest start date for activity C) so its latest start is week 2 (8 – 6).
- The latest start date for the project start is the earliest of the latest start dates for activities A, B and F. This is week zero. This is, of course, not very surprising since it tells us that if the project does not start on time it won't finish on time.

## 6.12 Identifying the Critical Path

The critical path is the longest path through the network.

There will be at least one path through the network (that is, one set of successive activities) that defines the duration of the project. This is known as the *critical path*. Any delay to any activity on this critical path will delay the completion of the project.

This float is also known as total float to distinguish it from other forms of float – see Section 6.13.

The difference between an activity's earliest start date and its latest start date (or, equally, the difference between its earliest and latest finish dates) is known as the activity's *float* – it is a measure of how much the start or completion of an activity may be delayed without affecting the end date of the project. Any activity with a float of zero is critical in the sense that any delay in carrying out the activity will delay the completion date of the project as a whole. There will always be at least one path through the network joining those critical activities – this path is known as the critical path and is shown bold in Figure 6.17.

The significance of the critical path is two-fold.

- In managing the project, we must pay particular attention to monitoring activities on the critical path so that the effects of any delay or resource unavailability are detected and corrected at the earliest opportunity.

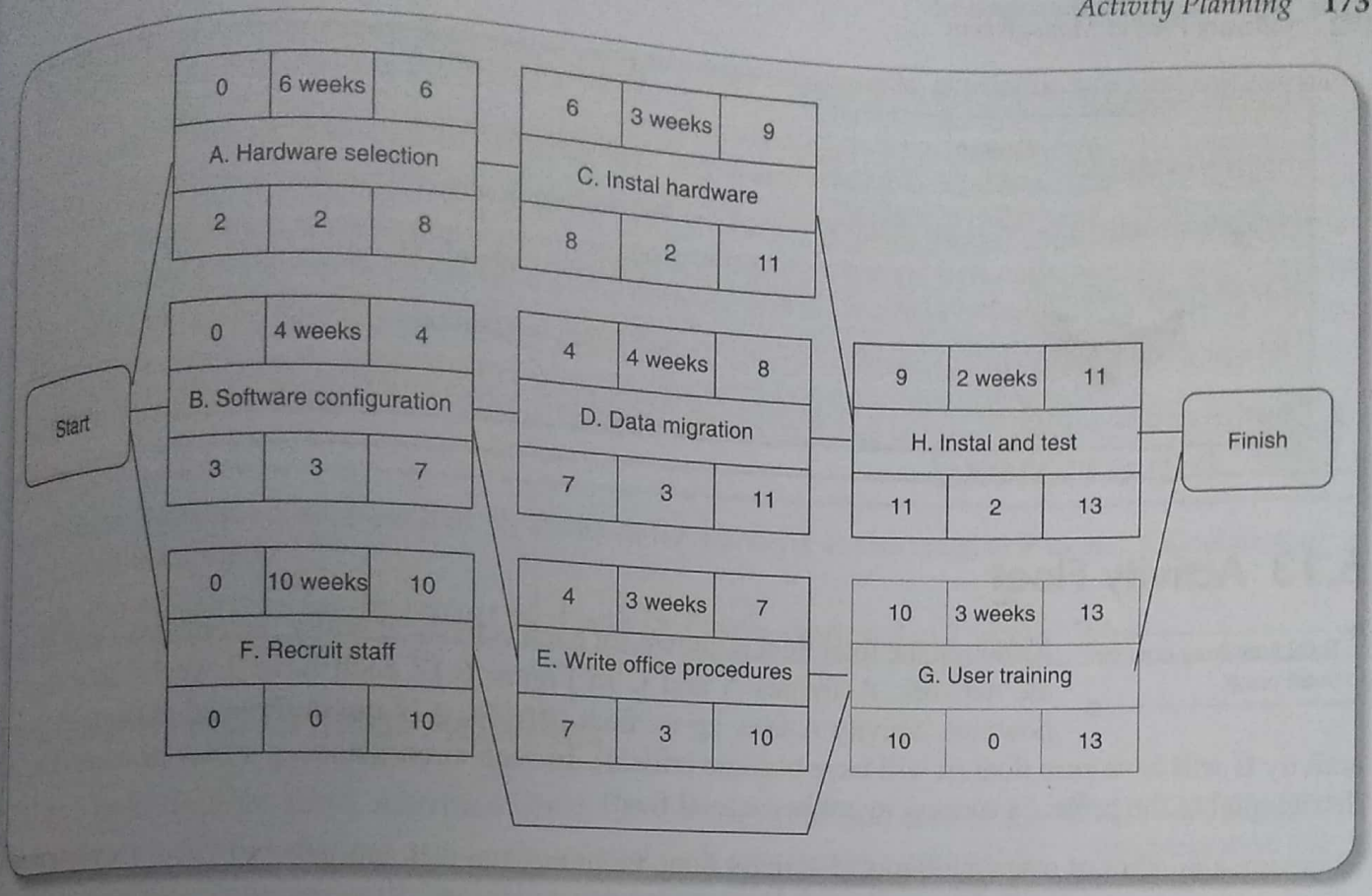



**FIGURE 6.17** The critical path

- In planning the project, it is the critical path that we must shorten if we are to reduce the overall duration of the project.

Figure 6.17 also shows the *activity span*. This is the difference between the earliest start date and the latest finish date and is a measure of the maximum time allowable for the activity. However, it is subject to the same conditions of interpretation as activity float, which is discussed in the next section.

## Exercise 6.2

Refer back to Amanda's CPM network illustrated in Figure 6.7.

Using the activity durations given in Table 6.2, calculate the earliest completion date for the project and identify the critical path on your network.

**TABLE 6.2** Estimated activity durations for Amanda's network

| Activity | Estimated duration (days) | Activity | Estimated duration (days) |
|---|---|---|---|
| Specify overall system | 34 | Design module C | 4 |
| Specify module A | 20 | Design module D | 4 |

(*Contd.*)

| | | | |
|---|---|---|---|
| Specify module B | 15 | Code/test module A | 30 |
| Specify module C | 25 | Code/test module B | 28 |
| Specify module D | 15 | Code/test module C | 15 |
| Check specification | 2 | Code/test module D | 25 |
| Design module A | 7 | System integration | 6 |
| Design module B | 6 | | |

## 6.13 Activity Float

Total float may only be used once.

Although the total float is shown for each activity, it really 'belongs' to a path through the network. Activities A and C in Figure 6.17 each have 2 weeks' total float. If, however, activity A uses up its float (that is, it is not completed until week 8) then activity B will have zero float (it will have become critical). In such circumstances it may be misleading and detrimental to the project's success to publicize total float!

There are a number of other measures of activity float, including the following:

- *Free float:* The time by which an activity may be delayed without affecting any subsequent activity. It is calculated as the difference between the earliest completion date for the activity and the earliest start date of the succeeding activity. This might be considered a more satisfactory measure of float for publicizing to the staff involved in undertaking the activities.
- *Interfering float:* The difference between total float and free float. This is quite commonly used, particularly in association with the free float. Once the free float has been used (or if it is zero), the interfering float tells us by how much the activity may be delayed without delaying the project end date – even though it will delay the start of subsequent activities.

### Exercise 6.3

Calculate the free float and interfering float for each of the activities shown in the activity network (Figure 6.17).

## 6.14 Shortening the Project Duration

If we wish to shorten the overall duration of a project we would normally consider attempting to reduce activity durations. In many cases this can be done by applying more resources to the task – working overtime or procuring additional staff, for example. The critical path indicates where we must look to save time – if we are trying to bring forward the end date of the project, there is clearly no point in attempting to shorten non-critical activities. Referring to Figure 6.17, it can be seen that we could complete the project in week 12 by reducing the duration of activity F by one week (to 9 weeks).

As we reduce activity times along the critical path we must continually check for any new critical path emerging and redirect our attention where necessary.

There will come a point when we can no longer safely, or cost-effectively, reduce critical activity durations in an attempt to bring forward the project end date. Further savings, if needed, must be sought in a consideration of our work methods and by questioning the logical sequencing of activities. Generally, time savings are to be found by increasing the amount of parallelism in the network and the removal of bottlenecks (subject always, of course, to resource and quality constraints).

## Exercise 6.4

Referring to Figure 6.17, suppose that the duration for activity F is shortened to 8 weeks. Calculate the end date for the project.

What would the end date for the project be if activity F were shortened to 7 weeks? Why?

## 6.15 Identifying Critical Activities

The critical path identifies those activities which are critical to the end date of the project; however, activities that are not on the critical path may become critical. As the project proceeds, activities will invariably use up some of their float and this will require a periodic recalculation of the network. As soon as the activities along a particular path use up their total float then that path will become a critical path and a number of hitherto non-critical activities will suddenly become critical.

For a more in-depth discussion of the role of the critical path in project monitoring, see Chapter 9.

It is therefore common practice to identify *near-critical* paths – those whose lengths are within, say, 10–20% of the duration of the critical path or those with a total float of less than, say, 10% of the project's uncompleted duration.

The importance of identifying critical and near-critical activities is that it is they that are most likely to be the cause of delays in completing the project. We shall see, in the next three chapters, that identifying these activities is an important step in risk analysis, resource allocation and project monitoring.

## 6.16 Activity-on-Arrow Networks

The developers of the CPM and PERT methods both originally used activity-on-arrow networks. Although now less common than activity-on-node networks, they are still used and introduce an additional useful concept – that of events. We will therefore take a brief look at how they are drawn and analysed using the same project example shown in Table 6.1.

In activity-on-arrow networks activities are represented by links (or arrows) and the nodes represent events of activities (or groups of activities) starting or finishing. Figure 6.18 illustrates our previous example (see Figure 6.14) drawn as an activity-on-arrow network.

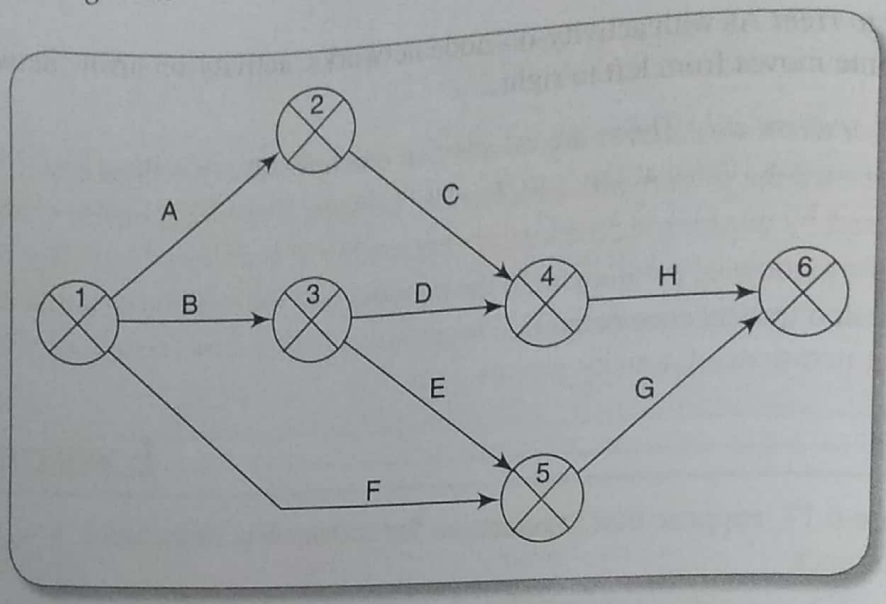



**FIGURE 6.18** An activity-on-arrow network

## Activity-on-arrow network rules and conventions

*A project network may have only one start node* This is a requirement of activity-on-arrow networks rather than merely desirable as is the case with activity-on-node networks.

*A project network may have only one end node* Again, this is a requirement for activity-on-arrow networks.

*A link has duration* A link represents an activity and, in general, activities take time to execute. Notice, however, that the network in Figure 6.18 does not contain any reference to durations. The links are not drawn in any way to represent the activity durations. The network drawing merely represents the logic of the project – the rules governing the order in which activities are to be carried out.

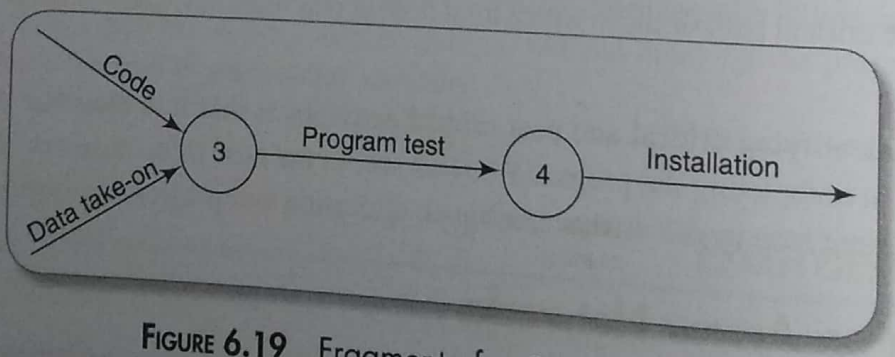



**FIGURE 6.19** Fragment of a CPM network

*Nodes have no duration* Nodes are events and, as such, are instantaneous points in time. The source node is the event of the project becoming ready to start and the sink node is the event of the project becoming completed. Intermediate nodes represent two simultaneous events – the event of all activities leading into a node having been completed and the event of all activities leading out of that node being in a position to be started.

In Figure 6.19, node 3 is the event that both 'coding' and 'data take-on' have been completed and activity 'program test' is free to start. 'Installation' may be started only when event 4 has been achieved, that is, as soon as 'program test' has been completed.

*Time moves from left to right* As with activity-on-node networks, activity-on-arrow networks are drawn, if at all possible, so that time moves from left to right.

*Nodes are numbered sequentially* There are no precise rules about node numbering but nodes should be numbered so that head nodes (those at the 'arrow' end of an activity) always have a higher number than tail events (those at the 'non-arrow' end of an activity). This convention makes it easy to spot loops.

*A network may not contain loops* Figure 6.20 demonstrates a loop in an activity-on-arrow network. As discussed in the context of precedence networks, loops are either an error of logic or a situation that must be resolved by itemizing iterations of activity groups.

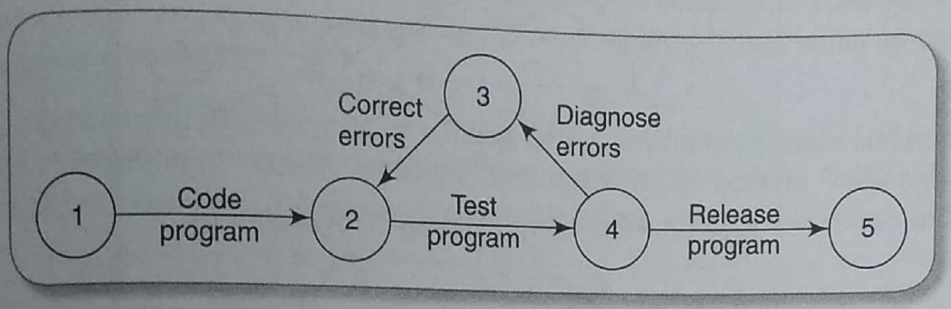



**FIGURE 6.20** A loop represents an impossible sequence

*A network may not contain dangles* A dangling activity, such as 'Write user manual' in Figure 6.21, cannot exist, as it would suggest there are two completion points for the project. If, in Figure 6.21, node 5 represents the true project completion point and there are no activities dependent on activity 'Write user manual', then the network should be redrawn so that activity 'Write user manual' starts at node 2 and terminates at node 5 – in practice, we would need to insert a dummy activity between nodes 3 and 5. In other words, all events, except the first and the last, must have at least one activity entering them and at least one activity leaving them and all activities must start and end with an event.

Dangles are not allowed in activity-on-arrow networks.

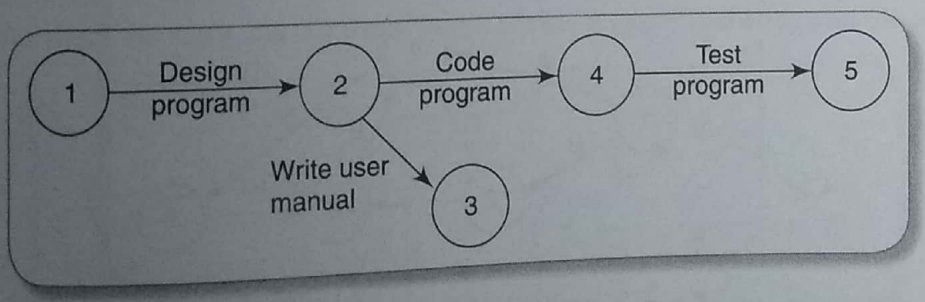



**FIGURE 6.21** A dangle

## Exercise 6.5

Take a look at the networks in Figure 6.22. State what is wrong with each of them and, where possible, redraw them correctly.

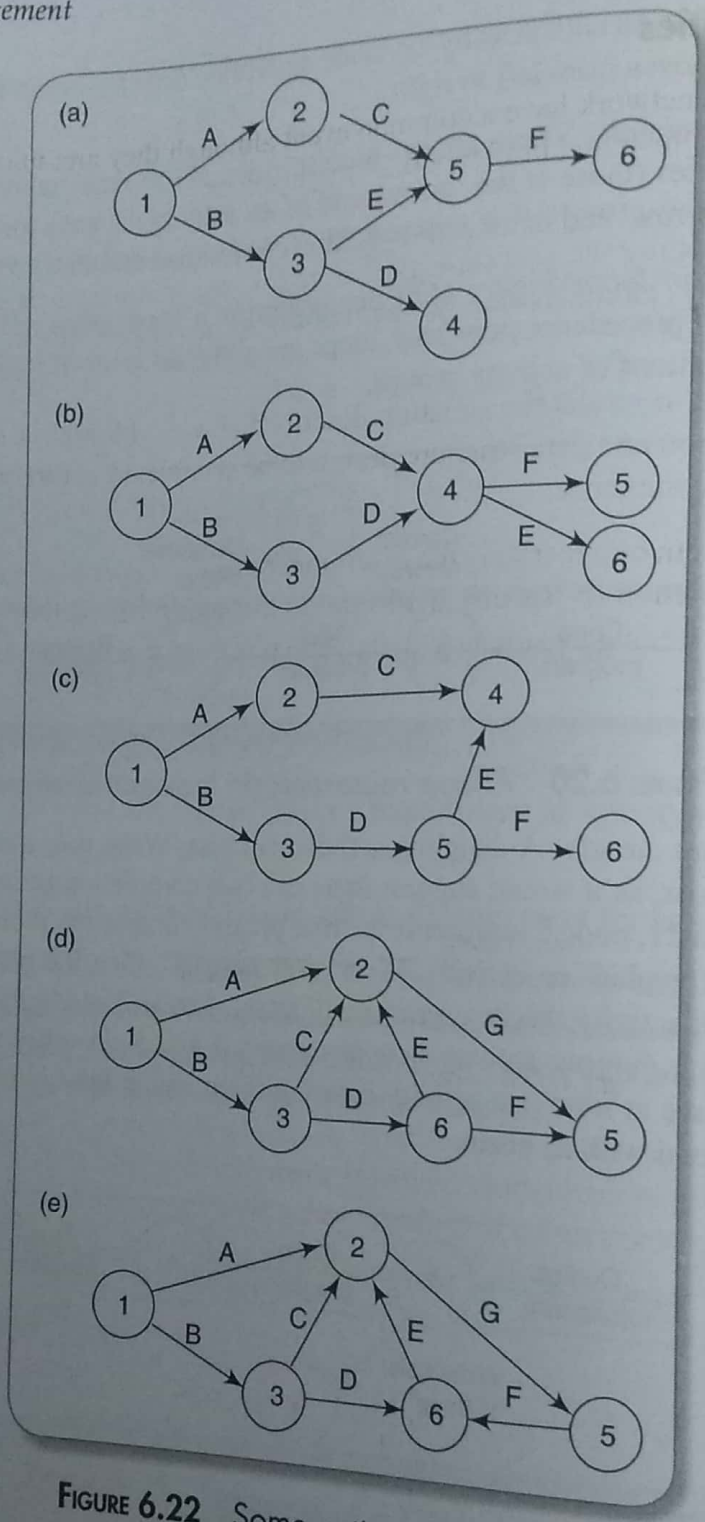



**FIGURE 6.22** Some activity networks

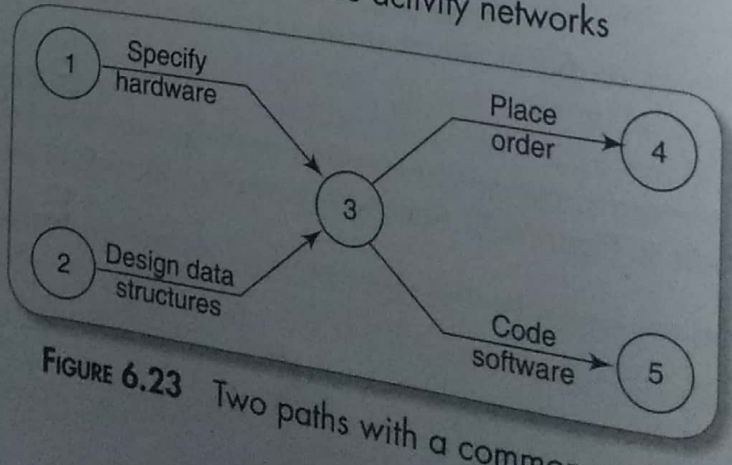



**FIGURE 6.23** Two paths with a common node

## Using dummy activities

When two paths within a network have a common event although they are, in other respects, independent, a logical error such as that illustrated in Figure 6.23 might occur.

Suppose that, in a particular project, it is necessary to specify a certain piece of hardware before placing an order for it and before coding the software. Before coding the software it is also necessary to specify the appropriate data structures, although clearly we do not need to wait for this to be done before the hardware is ordered.

Figure 6.23 is an attempt to model the situation described above, although it is incorrect in that it requires both hardware specification and data structure design to be completed before either an order may be placed or software coding may commence.

We can resolve this problem by separating the two (more or less) independent paths and introducing a dummy activity to link the completion of 'specify hardware' to the start of the activity 'code software'. This effectively breaks the link between data structure design and placing the order and is shown in Figure 6.24.

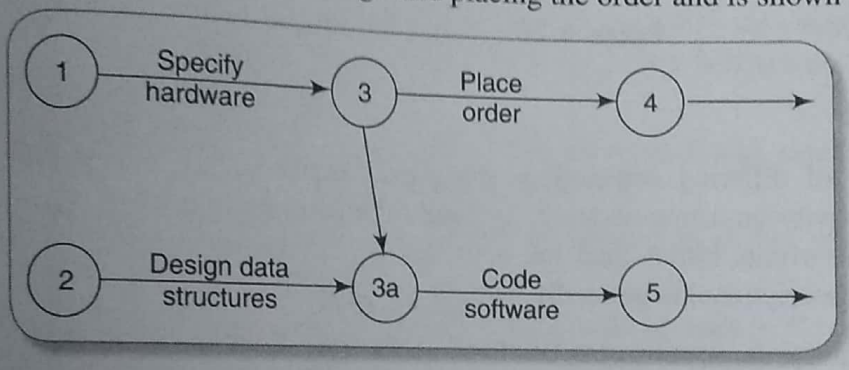



**FIGURE 6.24** Two paths linked by a dummy activity

Dummy activities, shown as dotted lines on the network diagram, have a zero duration and use no resources. They are often used to aid in the layout of network drawings as in Figure 6.25. The use of a dummy activity where two activities share the same start and end nodes makes it easier to distinguish the activity end-points.

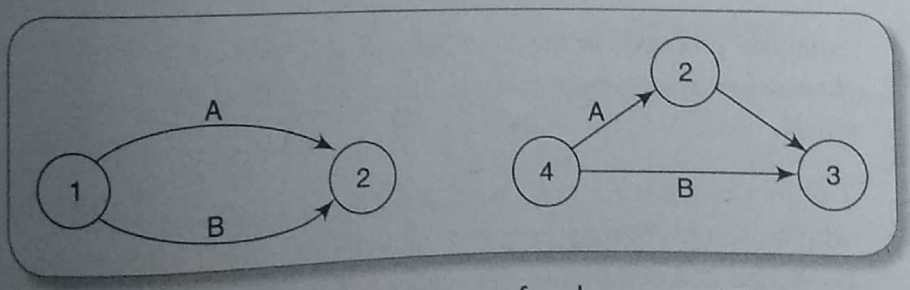



**FIGURE 6.25** Another use of a dummy activity

These are problems that do not occur with activity-on-node networks.

## Exercise 6.6

Take another look at Brigette's college payroll activity network fragment, which related to the earlier software selection process and which you developed in Exercise 3.4 (or take a look at the model answer in Figure B.2). Redraw this as an activity-on-arrow network.

## Representing lagged activities

Activity-on-arrow networks are less elegant when it comes to representing lagged parallel activities. We need to represent these with pairs of dummy activities as shown in Figure 6.26. Where the activities are lagged because a stage in one activity must be completed before the other may proceed, it is likely to be better to show each stage as a separate activity.

Where parallel activities have a time lag we may show this as a 'ladder' of activities: documentation may proceed alongside prototype testing so long as it starts at least a day later and will finish two days after the completion of prototype testing.

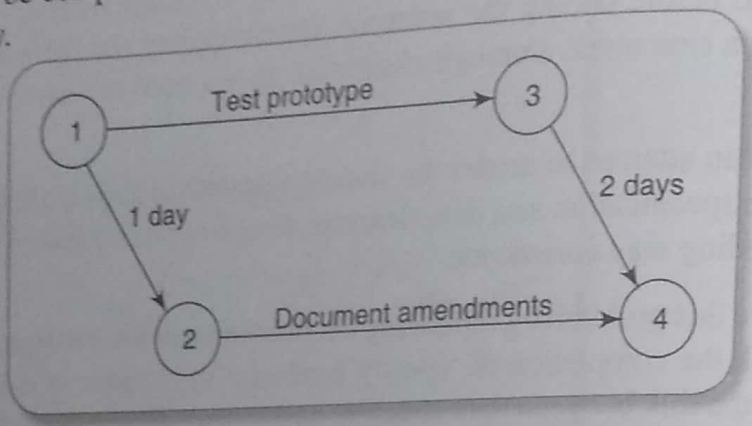
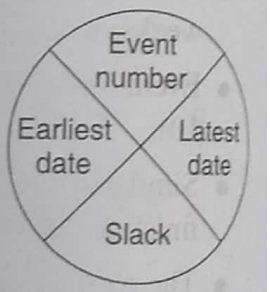



**FIGURE 6.26** Using the ladder technique to indicate lags

## Activity labelling

There are a number of differing conventions that have been adopted for entering information on an activity-on-arrow network. Typically the diagram is used to record information about the events rather than the activities – activity-based information (other than labels or descriptions) is generally held on a separate activity table.




One of the more common conventions for labelling nodes, and the one adopted here, is to divide the node circle into quadrants and use those quadrants to show the event number, the latest and earliest dates by which the event should occur, and the event slack (which will be explained later).

## Network analysis

During the forward pass, earliest dates are recorded as they are calculated. For events, they are recorded on the network diagram and for activities they are recorded on the activity table.

The forward pass rule: the earliest date for an event is the earliest finish date for all the activities terminating at that event. Where more than one activity terminates at a common event we take the latest of the earliest finish dates for those activities.

Analysis proceeds in the same way as with activity-on-node networks, although the discussion places emphasis on the events rather than activity start and completion times.

*The forward pass* The forward pass is carried out to calculate the earliest date on which each event may be achieved and the earliest dates on which each activity may be started and completed. The earliest date for an event is the earliest date by which all activities upon which it depends can be completed. Using Figure 6.18 and Table 6.1, the calculation proceeds according to the following reasoning.

- Activities A, B and F may start immediately, so the earliest date for event 1 is zero and the earliest start date for these three activities is also zero.
- Activity A will take 6 weeks, so the earliest it can finish is week 6 (recorded in the activity table). Therefore the earliest we can achieve event 2 is week 6.
- Activity B will take 4 weeks, so the earliest it can finish and the earliest we can achieve event 3 is week 4.

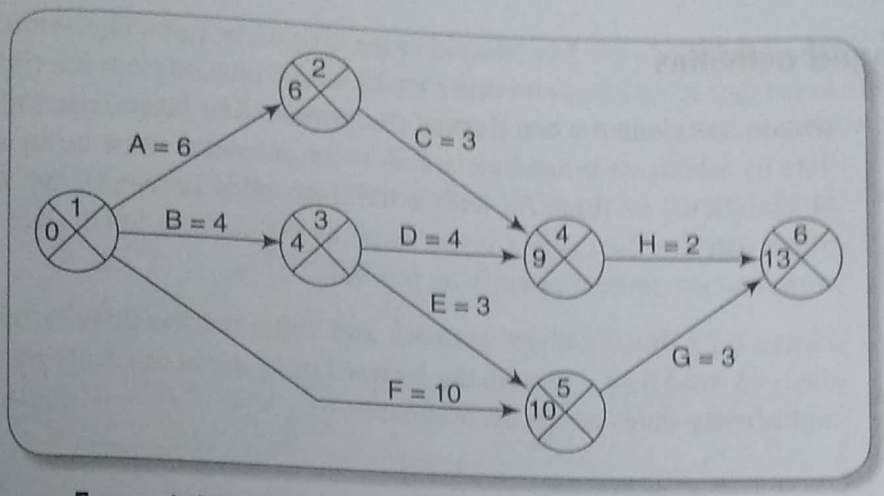



**FIGURE 6.27** A CPM network after the forward pass

- Activity F will take 10 weeks, so the earliest it can finish is week 10 – we cannot, however, tell whether or not this is also the earliest date that we can achieve event 5 since we have not, as yet, calculated when activity E will finish.
- Activity E can start as early as week 4 (the earliest date for event 3) and, since it is forecasted to take 3 weeks, will be completed, at the earliest, at the end of week 7.
- Event 5 may be achieved when both E and F have been completed, that is, week 10 (the later of 7 and 10).
- Similarly, we can reason that event 4 will have an earliest date of week 9. This is the later of the earliest finish for activity D (week 8) and the earliest finish for activity C (week 9).
- The earliest date for the completion of the project, event 6, is therefore the end of week 13 – the later of 11 (the earliest finish for H) and 13 (the earliest finish for G).

The results of the forward pass are shown in Figure 6.27 and Table 6.3.

**TABLE 6.3** Activity table after the forward pass

| Activity | Duration (weeks) | Earliest start date | Latest start date | Earliest finish date | Latest finish date | Total float |
|---|---|---|---|---|---|---|
| A | 6 | 0 | | 6 | | |
| B | 4 | 0 | | 4 | | |
| C | 3 | 6 | | 9 | | |
| D | 4 | 4 | | 8 | | |
| E | 3 | 4 | | 7 | | |
| F | 10 | 0 | | 10 | | |
| G | 3 | 10 | | 13 | | |
| H | 2 | 9 | | 11 | | |

The backward pass rule: the latest date for an event is the latest start date for all the activities that may commence from that event. Where more than one activity commences at a common event we take the earliest of the latest start dates for those activities.

*The backward pass* The second stage is to carry out a backward pass to calculate the latest date at which each event may be achieved, and each activity started and finished, without delaying the end date of the project. The latest date for an event is the latest date by which all immediately following activities must be started for the project to be completed on time. As with activity-on-node networks, we assume that the latest finish date for the project is the same as the earliest finish date – that is, we wish to complete the project as early as possible.

Figure 6.28 illustrates our network and Table 6.4 the activity table after carrying out the backward pass – as with the forward pass, event dates are recorded on the diagram and activity dates on the activity table.

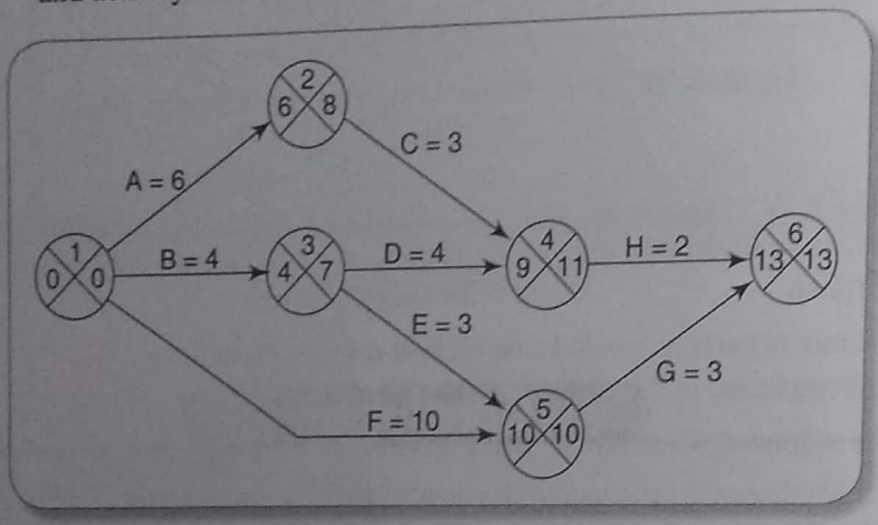


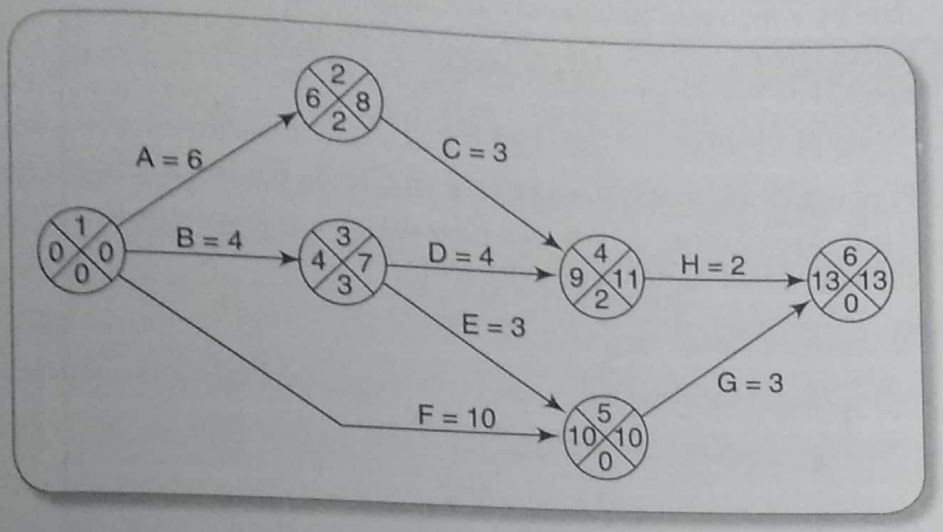

**FIGURE 6.28** CPM network after the backward pass

**TABLE 6.4** Activity table following the backward pass

| Activity | Duration (weeks) | Earliest start date | Latest start date | Earliest finish date | Latest finish date | Total float |
|---|---|---|---|---|---|---|
| A | 6 | 0 | 2 | 6 | 8 | |
| B | 4 | 0 | 3 | 4 | 7 | |
| C | 3 | 6 | 8 | 9 | 11 | |
| D | 4 | 4 | 7 | 8 | 11 | |
| E | 3 | 4 | 7 | 7 | 10 | |
| F | 10 | 0 | 0 | 10 | 10 | |
| G | 3 | 10 | 10 | 13 | 13 | |
| H | 2 | 9 | 11 | 11 | 13 | |

*Identifying the critical path* The critical path is identified in a way similar to that used in activity-on-node networks. We do, however, use a different concept, that of *slack*, in identifying the path. Slack is the difference

between the earliest date and the latest date for an event – it is a measure of how late an event may be without affecting the end date of the project. The critical path is the path joining all nodes with a zero slack (Figure 6.29).




The critical path is the longest path through the network.

**FIGURE 6.29** Critical path

## Conclusion

In this chapter, we have discussed the use of the critical path method and precedence networks to obtain an ideal activity plan. This plan tells us the order in which we should execute activities and the earliest and latest we can start and finish them.

These techniques help us to identify which activities are critical to meeting a target completion date.

In order to manage the project we need to turn the activity plan into a schedule that will specify precisely when each activity is scheduled to start and finish. Before we can do this, we must consider what resources will be required and whether or not they will be available at appropriate times. As we shall see, the allocation of resources to an activity may be affected by how we view the importance of the task and the risks associated with it. In the next two chapters we look at these aspects of project planning before we consider how we might publish a schedule for the project.

## Further Exercises

1. Draw an activity network using either activity-on-node or activity-on-arrow network conventions for each of the following projects:
   - Redecorating a room
   - Choosing and purchasing a desktop computer
   - Organizing and carrying out a survey of users' opinions of an information system
2. If you have access to a project planning application, use it to produce a project plan for the IOE annual maintenance contracts project. Base your plan on that used for Exercise 6.2 and verify that your application reports the same information as you calculated manually when you did the exercise.