SOFTWARE PROJECT MANAGEMENT
Hughes & Cotterell McGraw Hill

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 only finish 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 carried out.

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

Links normally have no duration Links represent the relationships between activities. In Figure 6.9 installation cannot start until program testing is complete. Program 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 until both 'Code' and 'Data take-on' have been completed and activity 'Install' cannot start until 'Program test' has finished. 'Code' and 'Data take-on' can therefore be said to be precedents of 'Program test', and 'Program test' is a precedent of 'Install'. Note that we do not speak of 'Code' and 'Data take-on' as precedents of 'Install' – that relationship is implicit in the previous statement.

Add arrows.

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.

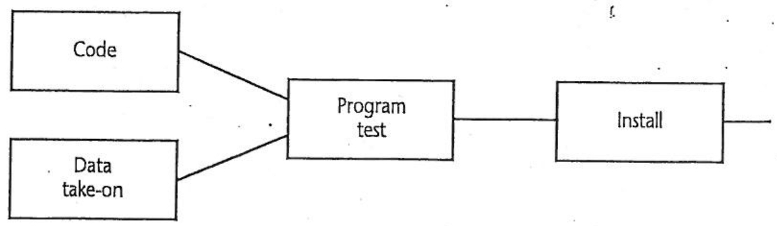


Figure 6.9 Fragment of a precedence network

ISBN-13 978-0-07-710989-9

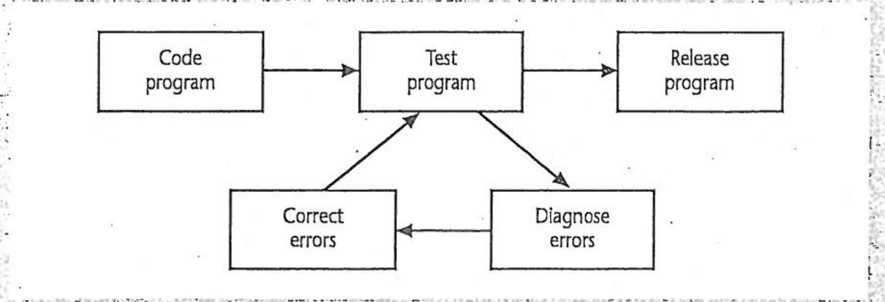


Figure 6.10 A loop represents an impossible sequence

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.

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

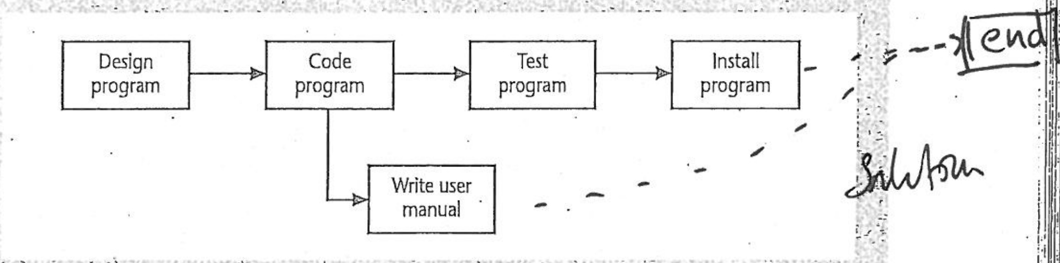


Figure 6.11 A dangle

Sign off = end of project.

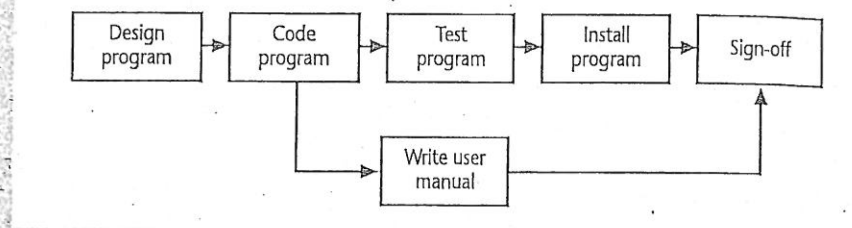


Figure 6.12 Resolving the dangle

representation of what should happen. The redrawn network is shown in Figure 6.12.

Representing lagged activities

We might come across situations where we wished 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 was 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.

Can forget

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

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

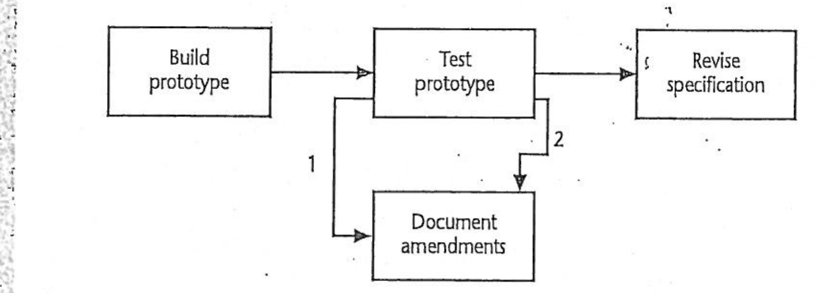


Figure 6.13 Indicating lags

costs or other resources that will be incurred or used at a constant rate over the duration of a set of activities.

Labelling conventions

Chose what software/company uses...

<table>
<tr><td colspan="2">Activity label</td><td>Duration</td></tr>
<tr><td>Earliest start</td><td rowspan="2">Activity description</td><td>Earliest finish</td></tr>
<tr><td>Latest start</td><td>Latest finish</td></tr>
<tr><td colspan="2">Activity span</td><td>Float</td></tr>
</table>

There are a number of differing conventions that have been adopted for entering information on an activity-on-node network. One of the more common conventions for labelling nodes, and the one adopted here, is shown on the left.

The activity label is usually a code developed to uniquely identify the activity and may incorporate a project code (for example, InvExt/P/3 to designate one of the programming activities for IOE's invoice extension 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.

6.9 Adding the time dimension

→ Asking programmers is best.
↪ go to the experts.

Should consult
Was about the programmers
↳ Always talk to users.

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.

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

Activity		*Duration (weeks)*	*Precedents*
A	Hardware selection	6	
B	Software design	4	
C	Install hardware	3	A
D	Code & test software	4	B
E	File take-on	3	B
F	Write user manuals	10	
G	User training	3	E, F
H	Install & test system	2	C, D

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

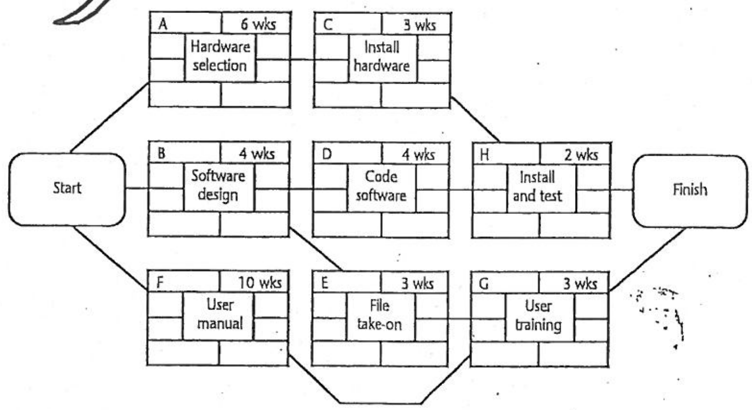


Figure 6.14 The precedence network for the example project

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.

6.10 The forward pass

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

Must be comprehensive.

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

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).

The forward pass and the calculation of earliest start dates is calculated 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

'he forward pass rule: he earliest start date or an activity is the :arliest finish date for he preceding activity. Vhere there is more han one immediately ›receding activity we ake the *latest* of the *:arliest finish dates* for hose activities.

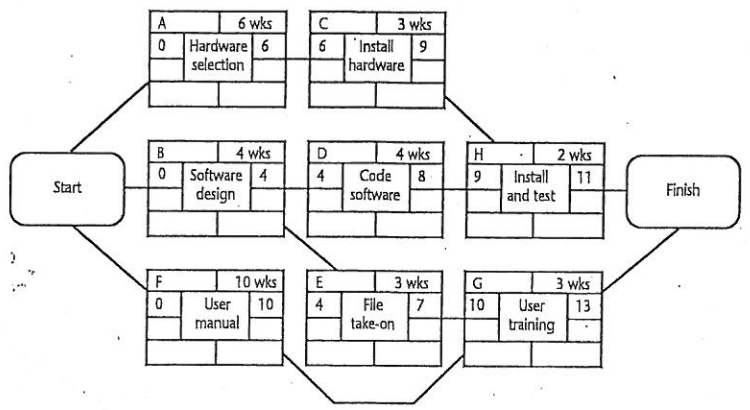


Figure 6.15 The network after the forward pass.

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 finished 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.

6.11 The 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. 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).

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

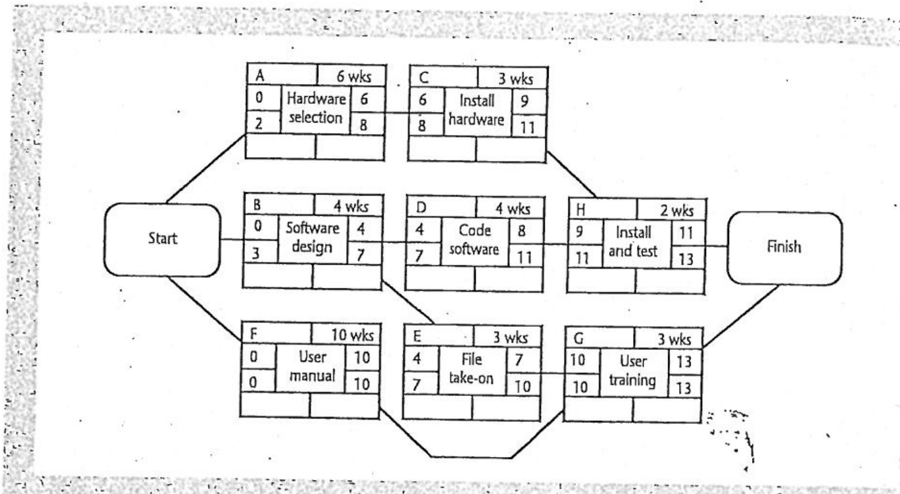


Figure 6.16 The network after the backward pass

- 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

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.

The critical path is the longest path through the network.

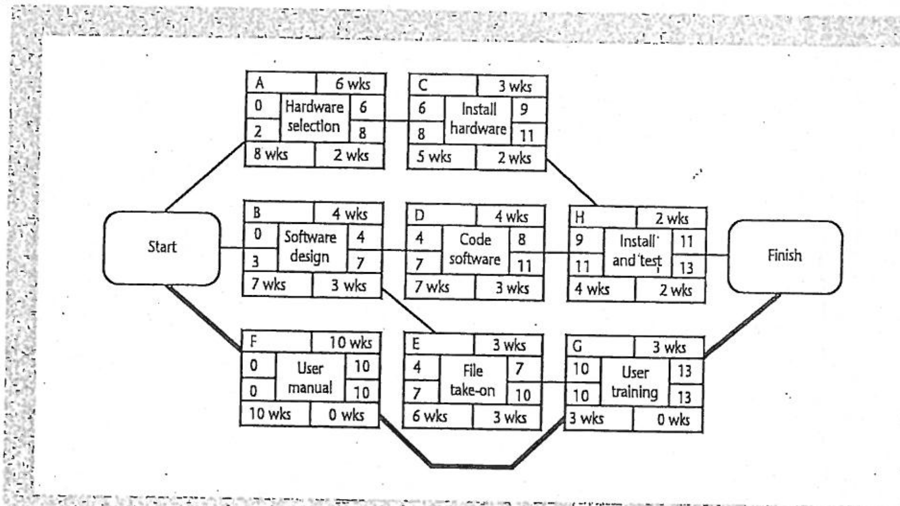


Figure 6.17 The critical path

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.
- 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.

6.13 Activity float

diff between Early & latest Start date.

Total float may only be used once.

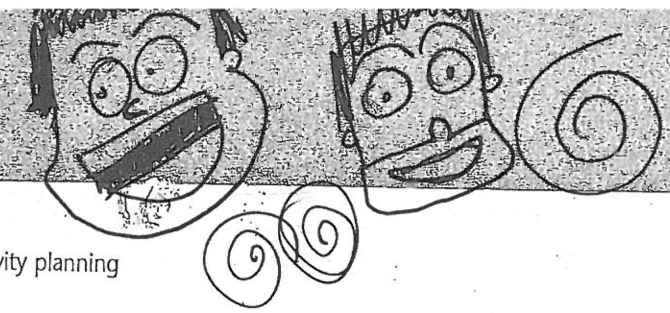
Although the total float is shown for each activity, it really 'belongs' to a path through the network. Activities A and C in Figure 6.16 each have 2 weeks' total float. If, however, activity A uses up its float (that is, it is not completed until week 8) then activity B will have zero float (it will 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

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
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		



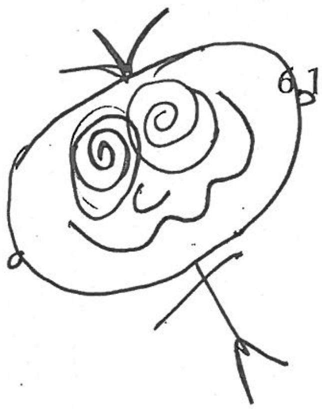
Dont Need Free or Interfering float

-1

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).

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?

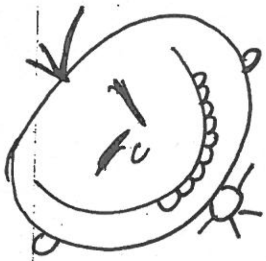
As we reduce activity times along the critical path we must continually check for any new critical path emerging and redirect our attention where necessary.

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).

6.15 Identifying critical activities

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

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.



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.

DO NOT NEED

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 drawn as 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 rep-

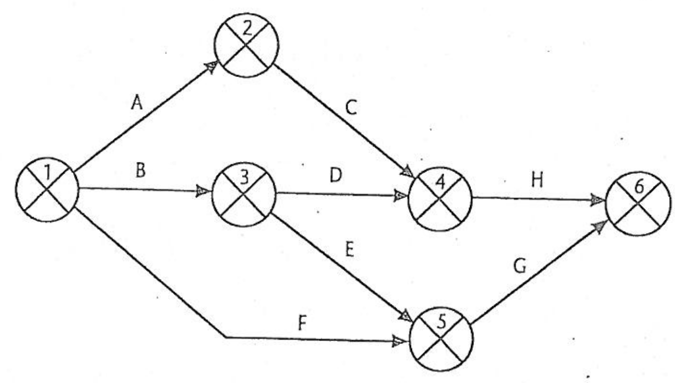


Figure 6.18 An activity-on-arrow network