Unit III

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.

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.

Chapter 11 discusses motivation in more detail.

- *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.

This coordination will normally form part of Programme Management.

- *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.

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.

Scheduling

'Time is nature's way of stopping everything happening at once'

Having

- worked out a method of doing the project
- identified the tasks to be carried
- assessed the time needed to do each task

need to allocate dates/times for the start and end of each activity

Activity networks

These help us to:

- Assess the feasibility of the planned project completion date
- Identify when resources will need to be deployed to activities
- Calculate when costs will be incurred

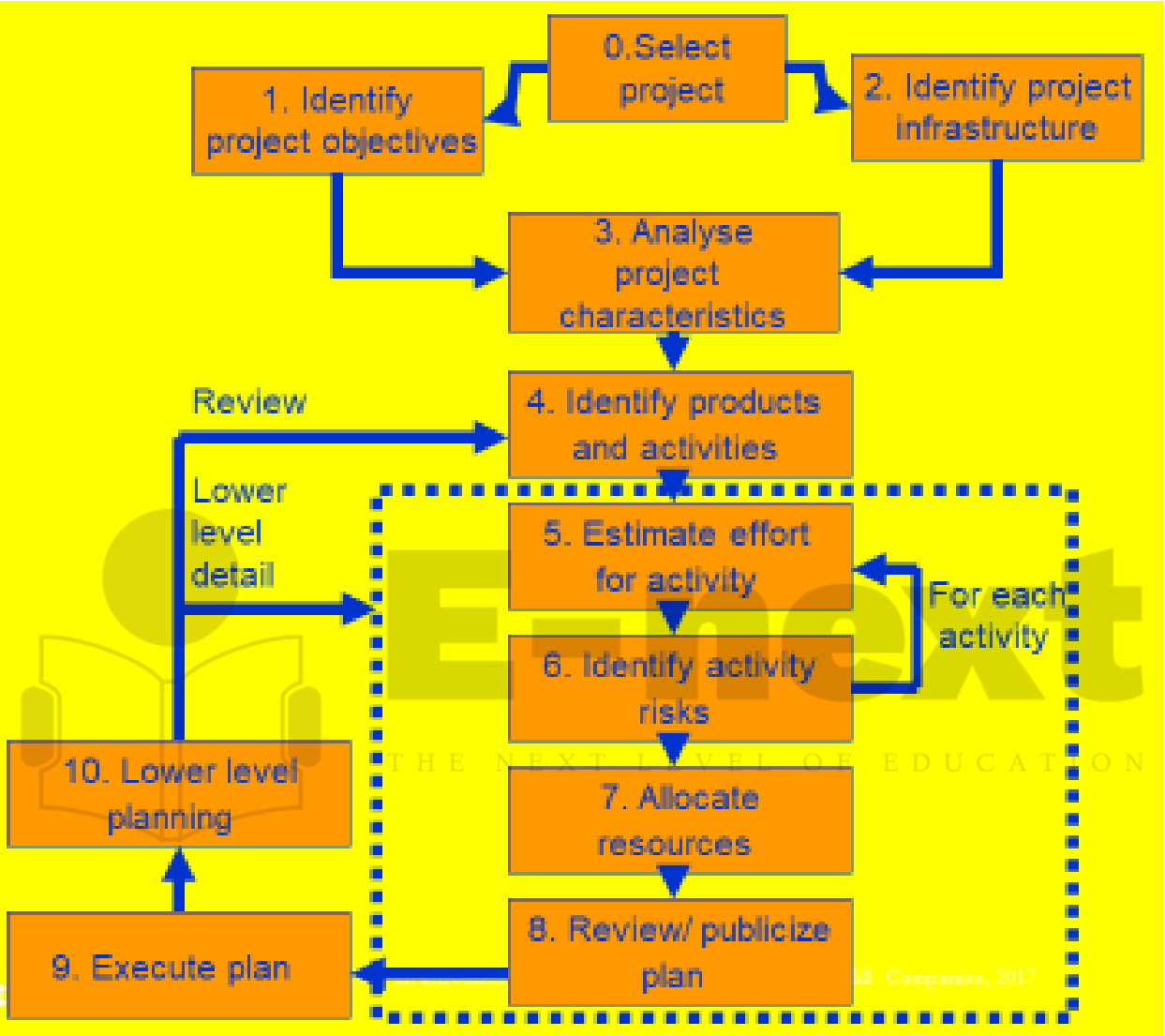
This helps the co-ordination and motivation of the project team

Defining activities

Activity networks are based on some assumptions:

- A project is:
  - Composed of a number of activities
  - May start when at least one of its activities is ready to start
  - Completed when all its activities are completed
- An activity
  - Must have clearly defined start and end-points
  - Must have resource requirements that can be forecast: these are assumed to be constant throughout the project

- Must have a duration that can be forecast
- May be dependent on other activities being completed first (precedence networks)



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.*

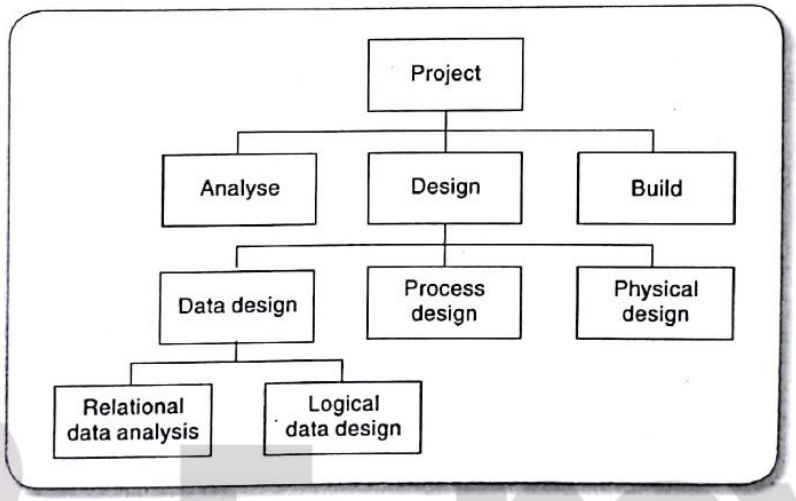


FIGURE 6.2 A fragment of an activity-based Work Breakdown Structure

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.

- **Work-based:** draw-up a Work Breakdown Structure listing the work items needed
- **Product-based approach**
  - list the deliverable and intermediate products of project – product breakdown structure (PBS)
  - Identify the order in which products have to be created
  - work out the activities needed to create the products

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.

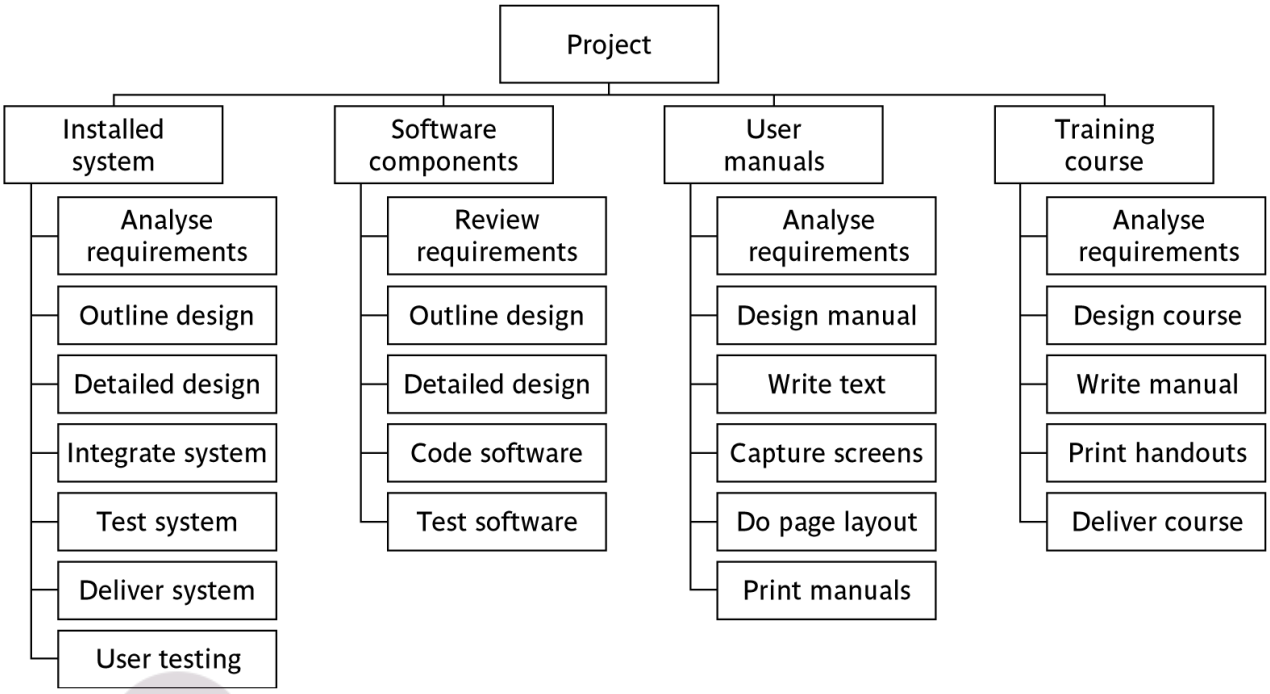
See I. Jacobson, G. Booch and J. Rumbaugh (1999) *The Unified Software Development Process*, Addison-Wesley.

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

The IBM MITP approach suggested the following 5 levels

- Level 1: Project
- Level 2: Deliverables
- Level 3: Components – which are key work items needed to produce the deliverables
- Level 4: Work packages: groups of tasks needed to produce the components
- Level 5: Tasks



The final outcome of the planning process

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.

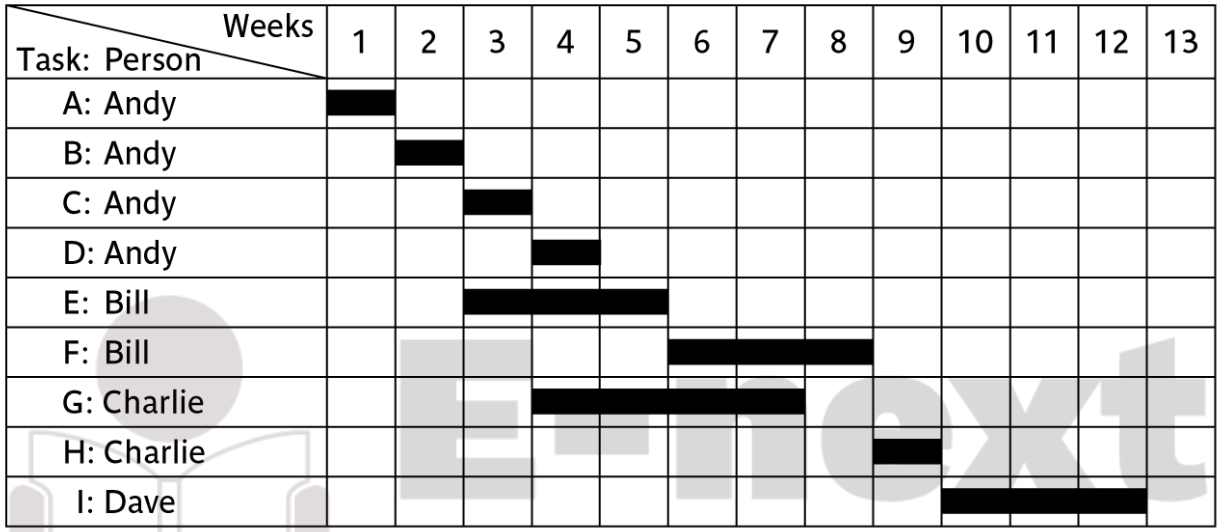
The chart shown has been drawn up taking account of the nature of the development process (that is, certain tasks must be completed before others may start) and the resources that are available (for example, activity C follows activity B because Andy cannot work on both tasks at the same time). In drawing 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.



Activity key

A: Overall design
B: Specify module 1
C: Specify module 2
D: Specify module 3
E: Code module 1
F: Code module 3
G: Code module 2
H: Integration testing
I: System tesing

Fig 6.6. A project plan as a bar chart

*The chart tells us **who** is doing **what** and **when**.*

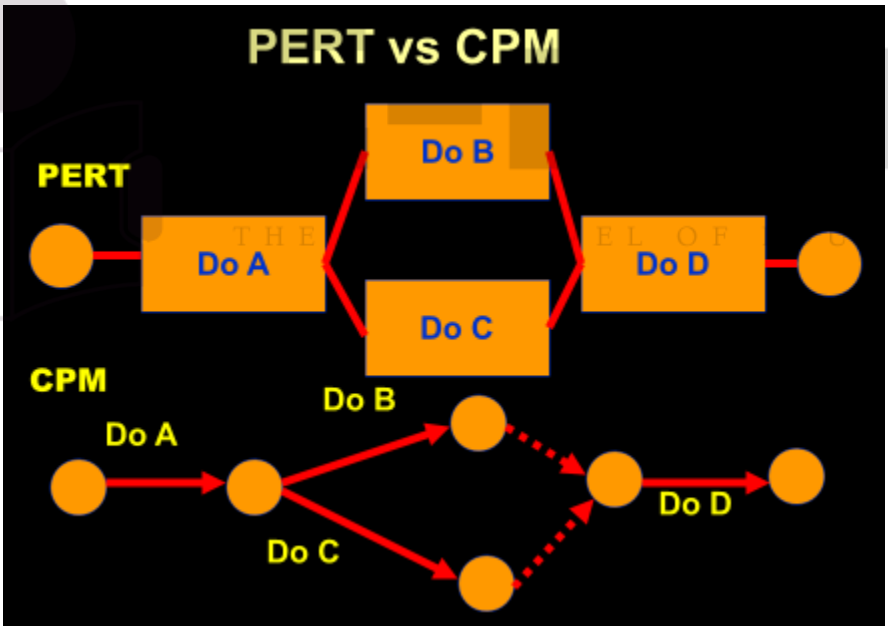
6.7 Network Planning Models

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.



PERT was devised to support the development of the Polaris missile in the late 1950's. CPM was developed by Du Pont Chemical Company who published the method in 1958.

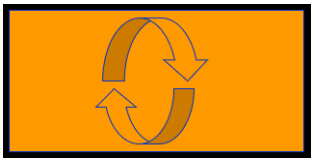
PERT is an activity-on-node notation – the 'nodes' are the boxes which represent activities

CPM uses an activity-on-arrow notation where the arrows are the activities.

The approached described here is based on PERT but the other approach is described in the textbook as well – see Section 6.16

Drawing up a PERT diagram

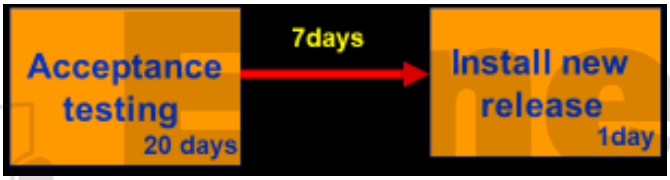
- No looping back is allowed – deal with iterations by hiding them within single activities



- *milestones* – 'activities', such as the start and end of the project, which indicate transition points. They have zero duration.

Lagged activities

- where there is a fixed delay between activities e.g. seven days notice has to be given to users that a new release has been signed off and is to be installed



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.

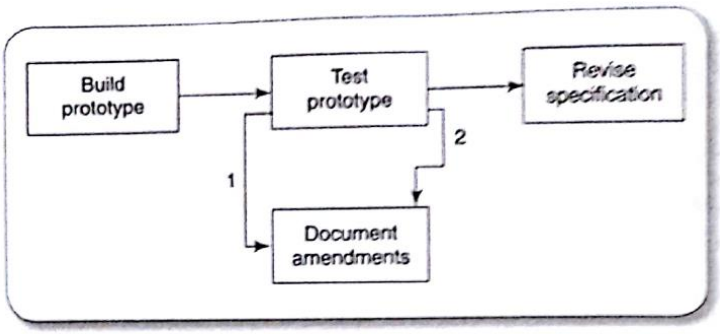


FIGURE 6.13 Indicating lags

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.

Types of links between activities

Finish to start: The following activity starts when the previous one has been finished

e.g. testing starts when coding has been completed

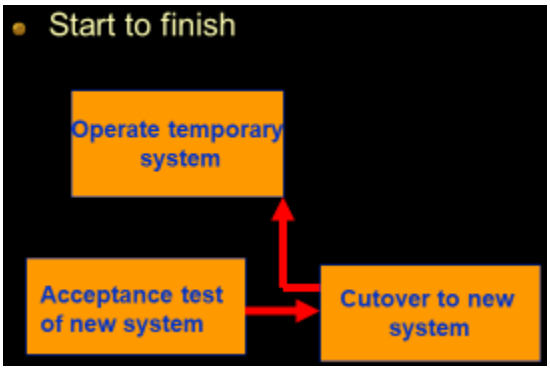
Start to start: When one activity starts another has to start as well

e.g. when prototype testing starts amendment documentation has to start as well

Finish to finish: when one activity finishes the other must finish too

e.g. when the testing of the prototype is completed so is the documentation of any amendments

You could use these with lags e.g. documentation of the changes to the prototype starts 1 day after the testing and finishes 2 days after testing has been completed



Start to finish – in the example when the cutover to the new system takes place, the operation of the temporary system is no longer needed. Although the cutover depends of the acceptance testing to be completed, the implication is that the cutover might not start straight after acceptance testing.

Start and finish times

- Activity 'write report software'
- Earliest start (ES)
- Earliest finish (EF) = ES + duration
- Latest finish (LF) = latest task can be completed without affecting project end Latest start = LF - duration

- The time that an activity can start depends on its relationship with the other tasks in the project. The earliest start is when the earliest of the preceding activities upon which the current activity depends will be completed, so that the current one can start. If it starts at this time the earliest the current activity can finish is the earliest start plus its duration.
- However, it may be that the activity, although it *can* start, can be delayed because later activities do not have to start right way. This gives us a latest finish date. The latest start date is the latest finish date less the duration of the activity.
- When a student is given coursework to do they do not necessarily start it straight away. They might note when it has got to be handed in, work out that it will only take about three days to do - with a bit of luck - and wait until three days before the hand-in before they start.

Example

- earliest start = day 5
- latest finish = day 30
- duration = 10 days
- earliest finish = ?
- latest start = ?

Float = LF - ES – duration

What is it in this case?

The earliest finish (EF) would be day 5 plus 10 days i.e. day 15.

The latest start (LS) would be day 30 – 10 days i.e. day 20

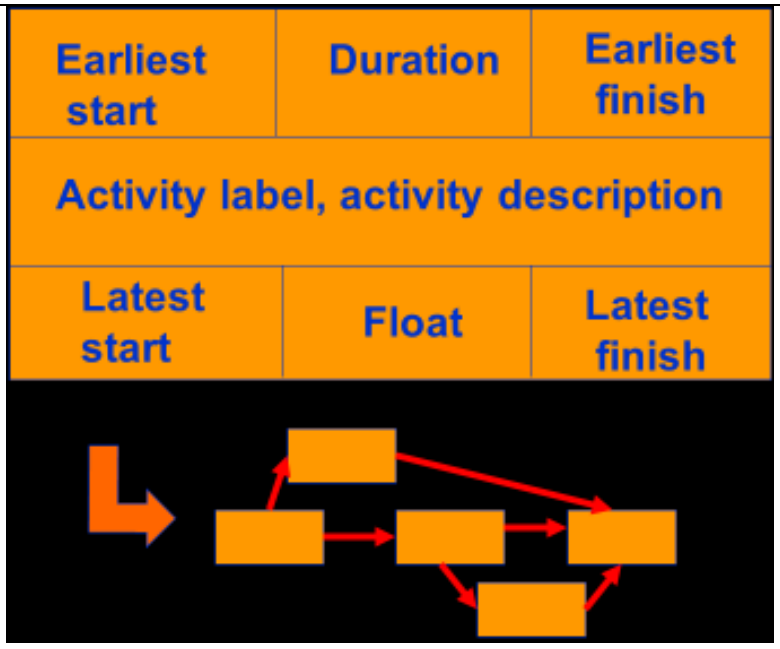
The float would be 30 – 5 – 10 = 15 days

This also is the same as LF – EF or LS - ES

'Day 0'

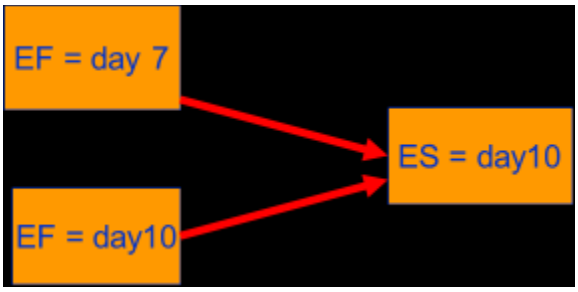
- Note that in the last example, day numbers used rather than actual dates
- Makes initial calculations easier – not concerned with week-ends and public holidays
- For **finish** date/times Day 1 means at the END of Day 1.
- For a **start** date/time Day 1 also means at the END of Day 1.
- The first activity therefore begin at Day 0 i.e. the end of Day 0 i.e. the start of Day 1

- This also means activity A could finish on Day 5, for example, and a dependent activity B could then start on Day 5 as well (not Day 6).
- All this may make more sense if you think about an activities that finish and start halfway through a particular day.
- Although we talk about 'Days' other time units could be equally valid e.g. Hours, weeks, months. In case of weeks point out that we usually assume 5 day working weeks in our calculations – unless otherwise specified.

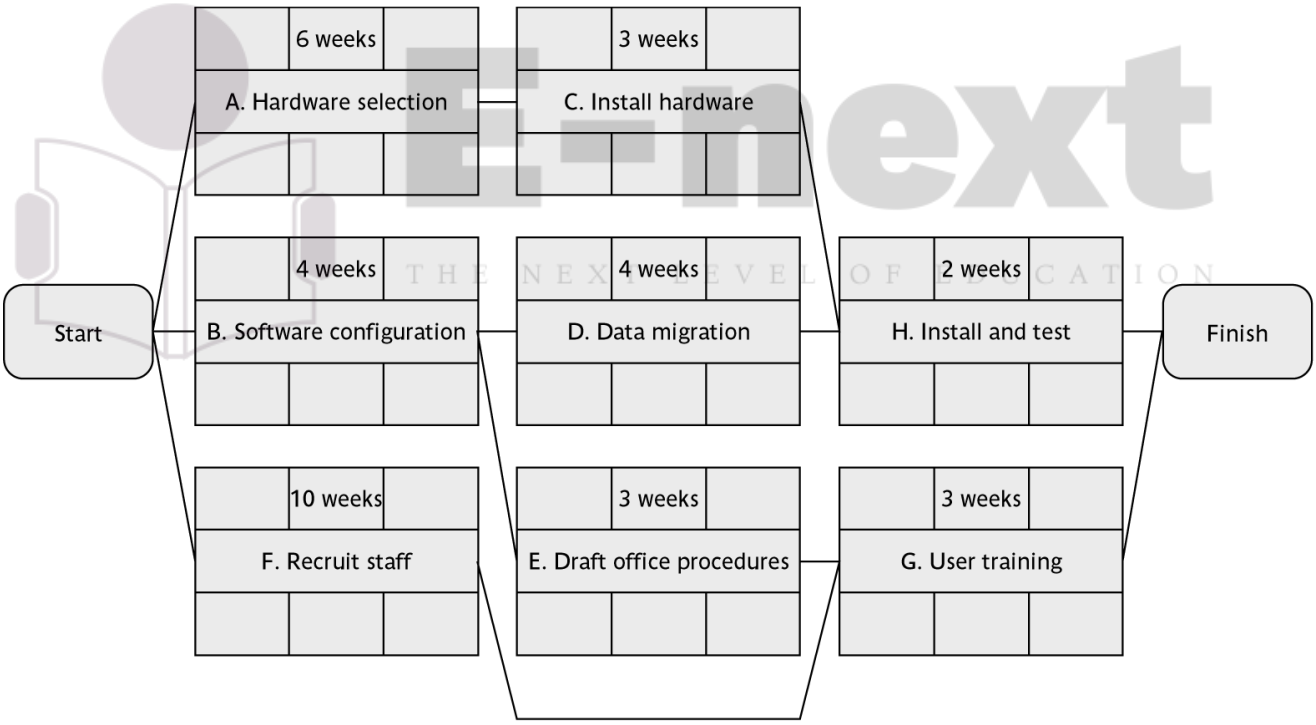


Forward pass

- Start at beginning (Day 0) and work forward following chains.
- Earliest start date for the *current* activity = earliest finish date for the *previous*
- When there is more than one previous activity, take the *latest* earliest finish



Example of an activity network

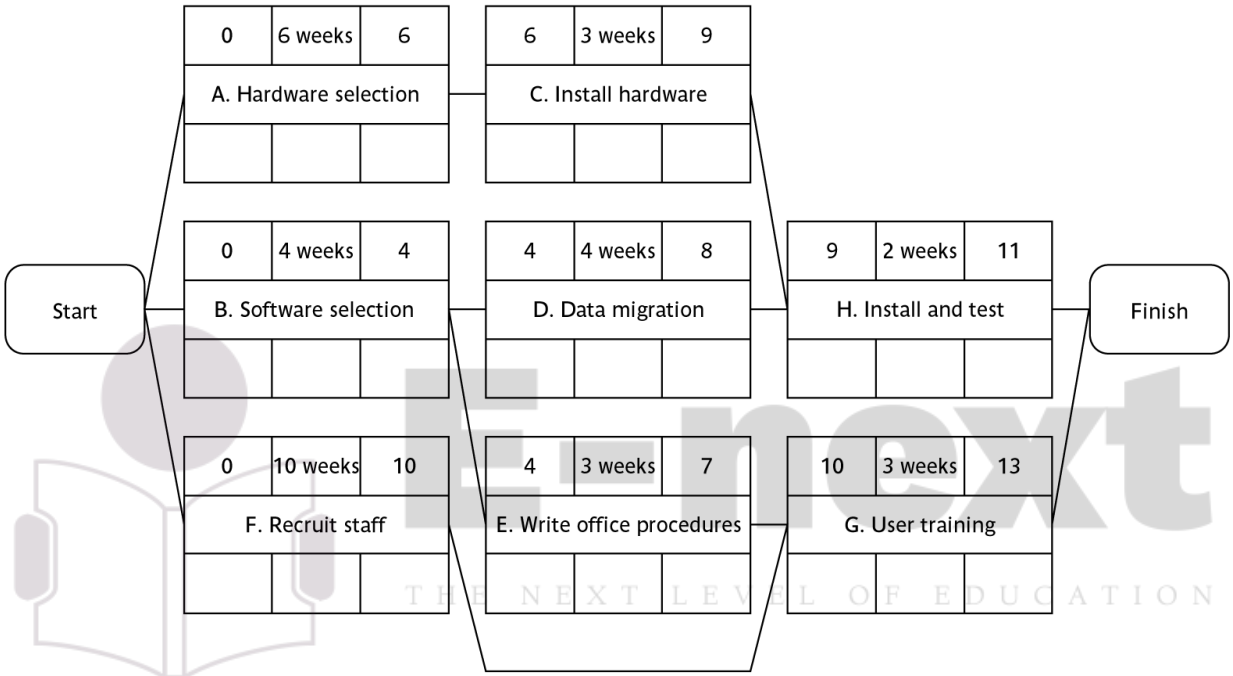


Backward pass

- Start from the *last* activity
- Latest finish (LF) for last activity = earliest finish (EF)
- work backwards

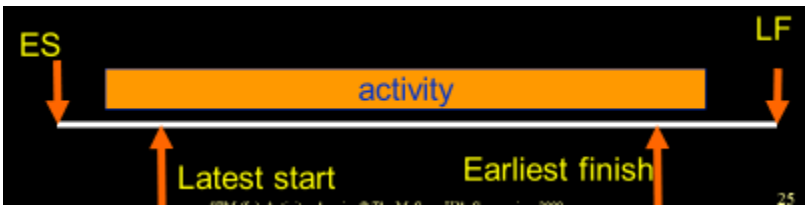
- Latest finish for *current* activity = Latest start for the *following*
- More than one following activity - take the *earliest* LS
- Latest start (LS) = LF for activity - duration

Example: LS for all activities?



Float

Float = Latest finish - Earliest start - Duration



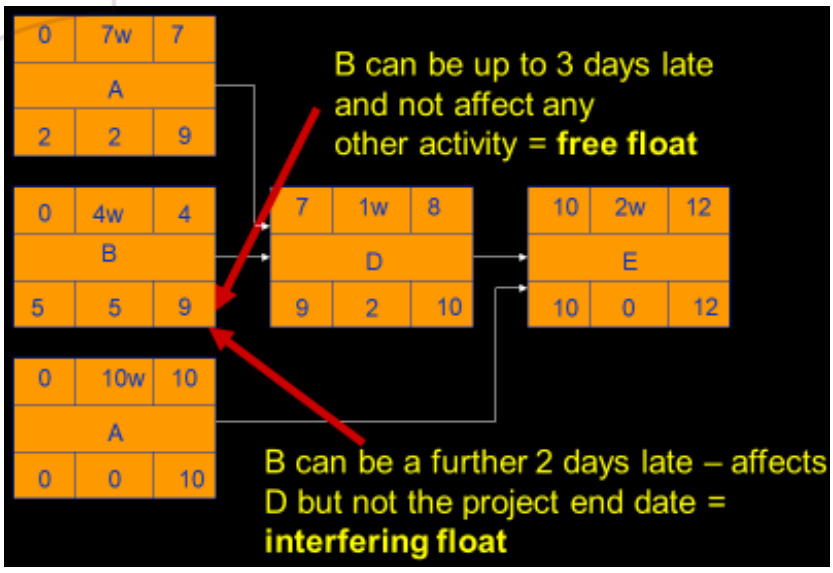
Note that Float can also be calculated as the difference between the earliest and latest start dates for an activity *or* the difference between the earliest and latest finish dates.

Critical path

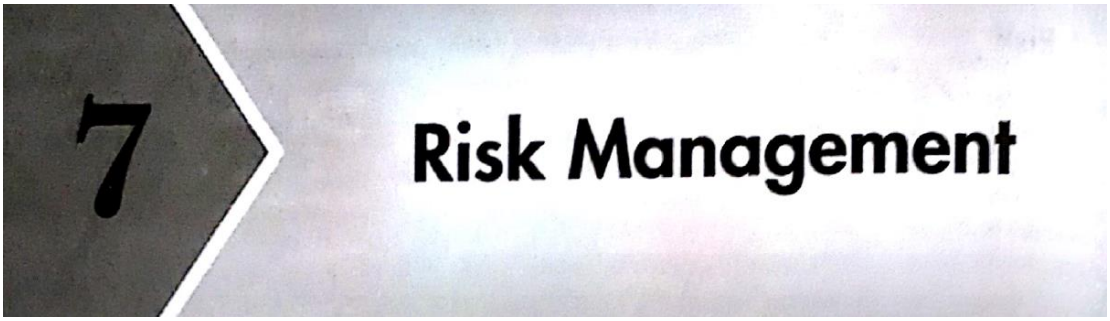
- Note the path through network with zero floats
- Critical path: any delay in an activity on this path will delay whole project
- Can there be more than one critical path?
- Can there be no critical path?
- Sub-critical paths

- Yes, there could be more than one critical path if the two longest paths through the network were of equal length.
- Where the target completion date for the project was imposed rather than calculated from the earliest finish dates, it might be later that the earliest finish date. In this case there would be no chains of activities with zero floats
- The durations of activities are only estimates to start with. As the project proceeds, the estimates will be replaced by actual durations which could be different. This could change the sequence of activities identified as the critical path. Sub-critical paths are chains of activities, not on the planned critical path, but which have small floats and which could easily become the critical path as the project develops.

Free and interfering float



Total float = LF – ES – duration (or LS-ES or LF-EF)
Free float = ES for following activity less EF for the current
Interfering float = total float – free float



Risk management

Learning Objectives:

- Definition of 'risk' and 'risk management'
- Some ways of categorizing risk
- Risk management
  - Risk identification – what are the risks to a project?
  - Risk analysis – which ones are really serious?
  - Risk planning – what shall we do?
  - Risk monitoring – has the planning worked?
- We will also look at PERT risk and critical chains

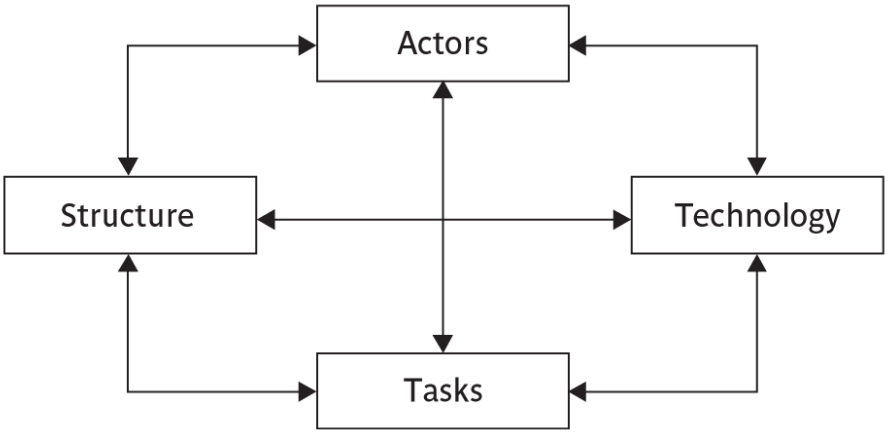
Some definitions of risk

'the chance of exposure to the adverse consequences of future events' PRINCE2

'an uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives' PM-BOK

- Risks relate to **possible future** problems, not current ones
- They involve a possible cause and its effect(s) e.g. developer leaves > task delayed

Categories of risk



This is based on Lyytinen's sociotechnical model of risk

- **Actors** relate to all those involved in the project including both developers, users and managers e.g. a risk could be that high staff turnover leads to information of importance to the project being lost
- **Technology** – both that used to implement the project and that embedded in the project deliverables – risk could be that the technologies selected are not in fact appropriate.
- **Structure** – this includes management procedures, risk here is that a group who need to carry out a particular project task are not informed of this need because they are not part of the project communication network
- **Tasks** – the work to be carried out. A typical risk is that the amount of effort needed to carry out the task is underestimated.

A risk could be well belong to more than one of the four areas – for example, estimates being wrong could be influenced by problems with actors (e.g. lack of experience with a technical domain) or the structure (over optimism of managers keen to win work).

Risk Management Approaches

- **Proactive:**
  - The proactive approaches try to anticipate the possible risks that the project is susceptible to.
  - After identifying the possible risks, actions are taken to eliminate the risks.

- **Reactive:**
  - Reactive approaches take no action until an unfavourable event occurs.
  - Once an unfavourable event occurs, these approaches try to contain the adverse effects associated with the risk and take steps to prevent future occurrence of the same risk events.

A framework for dealing with risk

The planning for risk includes these steps:

- Risk identification – what risks might there be?
- Risk analysis and prioritization – which are the most serious risks?
- Risk planning – what are we going to do about them?
- Risk monitoring – what is the current state of the risk?

Risk identification

Approaches to identifying risks include:

- **Use of checklists** – usually based on the experience of past projects
- **Brainstorming** – getting knowledgeable stakeholders together to pool concerns
- **Causal mapping** – identifying possible chains of cause and effect

Boehm's top 10 development risks

Barry Boehm surveyed software engineering project leaders to find out the main risks that they had experienced with their projects. For each risk, some risk reduction techniques has been suggested.

7.6 Risk Identification

The two main approaches to the identification of risks are the use of *checklists* and *brainstorming*.

Checklists are simply lists of the risks that have been found to occur regularly in software development projects. A specialized list of software development risks by Barry Boehm appears in Table 7.1 in a modified version. Ideally a group of representative project stakeholders examines a checklist identifying risks applicable to their project. Often the checklist suggests potential countermeasures for each risk.

TABLE 7.1 Software project risks and strategies for risk reduction

Risk	Risk reduction techniques
Personnel shortfalls	Staffing with top talent; job matching; teambuilding; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental development; recording and analysis of past projects; standardization of methods
Developing the wrong software functions	Improved software evaluation; formal specification methods; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement
Gold plating	Requirements scrubbing; prototyping; cost–benefit analysis; design to cost
Late changes to requirements	Stringent change control procedures; high change threshold; incremental development (deferring changes)
Shortfalls in externally supplied components	Benchmarking; inspections; formal specifications; contractual agreements; quality assurance procedures and certification
Shortfalls in externally performed tasks	Quality assurance procedures; competitive design or prototyping; contract incentives
Real-time performance shortfalls	Simulation; benchmarking; prototyping; tuning; technical analysis
Development technically too difficult	Technical analysis; cost–benefit analysis; prototyping; staff training and development

This top ten list of software risks is based on one presented by Barry Boehm in his *Tutorial on Software Risk Management*, IEE Computer Society, 1989.

Project management methodologies, such PRINCE2, often recommend that on completion of a project a review identifies any problems during the project and the steps that were (or should have been) taken to resolve or avoid them. These problems could in some cases be added to an organizational risk checklist for use with new projects.

The 'lessons learnt' report differs from a 'post implementation review' (PIR). It is written on project completion and focuses on project issues. A PIR, produced when the application has been operational for some time, focuses on business benefits.

'Gold plating' refers to inclusion of features that in fact are unnecessary and which end up never actually being used.

Risk prioritization

Risk exposure (RE) = (potential damage) x (probability of occurrence)

Ideally

Potential damage: a money value e.g. a flood would cause £0.5 millions of damage

Probability 0.00 (absolutely no chance) to 1.00 (absolutely certain) e.g. 0.01 (one in hundred chance)

RE = £0.5m x 0.01 = £5,000

Crudely analogous to the amount needed for an insurance premium

7.7 Risk Assessment

A common problem with risk identification is that a list of risks is potentially endless. A way is needed of distinguishing the damaging and likely risks. This can be done by estimating the *risk exposure* for each risk using the formula:

risk exposure = (potential damage) × (probability of occurrence)

Using the most rigorous – but not necessarily the most practical – approach, the potential damage would be assessed as a money value. Say a project depended on a data centre vulnerable to fire. It might be estimated that if a fire occurred a new computer configuration could be established for £500,000. It might also be estimated that where the computer is located there is a 1 in 1000 chance of a fire actually happening, that is a probability of 0.001.

The risk exposure in this case would be:

£500,000 × 0.001 = £500

A crude way of understanding this value is as the minimum sum an insurance company would require as a premium. If 1000 companies, all in the same position, each contributed £500 to a fund then, when the 1 in 1000 chance of the fire actually occurred, there would be enough money to cover the cost of recovery.

Risk probability: qualitative descriptors

Probability level	*Range*
High	Greater than 50% chance of happening
Significant	30-50% chance of happening
Moderate	10-29% chance of happening
Low	Less than 10% chance of happening

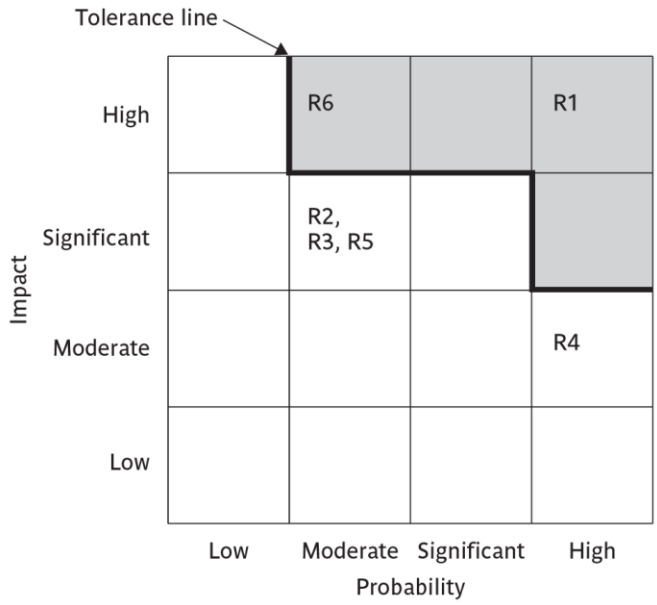
Qualitative descriptors of impact on cost and associated range values

Impact level	*Range*
High	Greater than 30% above budgeted expenditure
Significant	20 to 29% above budgeted expenditure
Moderate	10 to 19% above budgeted expenditure
Low	Within 10% of budgeted expenditure.

Similar tables can be produced for the impact on project duration and on the quality of project deliverables.

The problem with the qualitative approach is how do you combine the judgements about probability and impact – you can't multiply them together.

Probability impact matrix



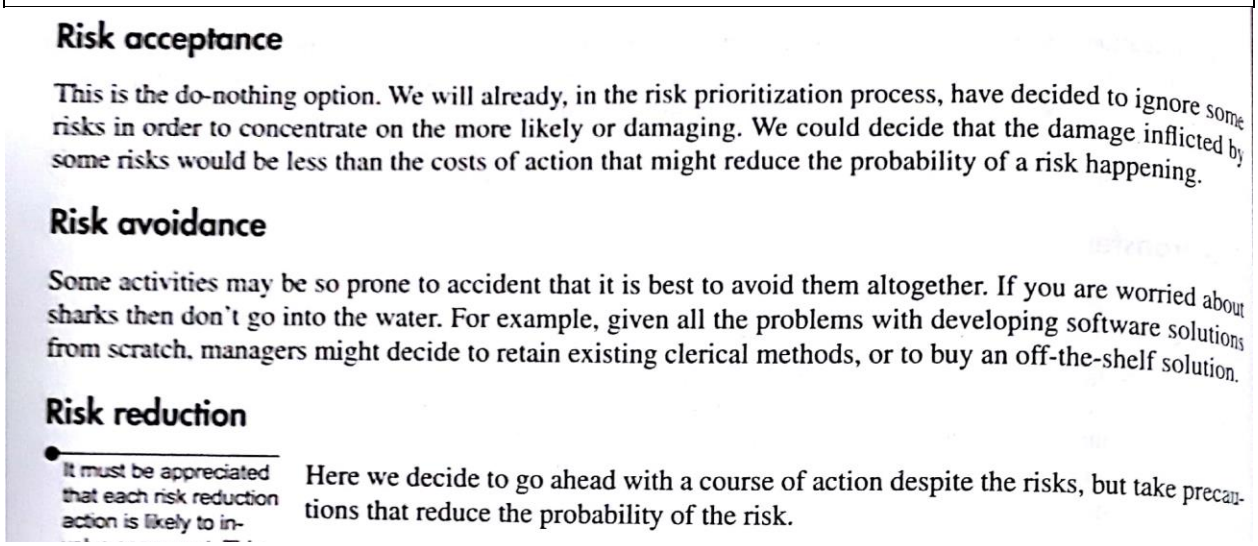
R1, R2 etc refer to particular risks. They are located on the grid according to the likelihood and impact ratings that have been allocated to them. A zone around the top right hand corner of the grid can be designated and risks falling within that zone are treated as requiring urgent action.

Risk planning

Risks can be dealt with by:

- Risk acceptance
- Risk avoidance
- Risk reduction
- Risk transfer
- Risk mitigation/contingency measures

- **Risk acceptance** – the cost of avoiding the risk may be greater than the actual cost of the damage that might be inflicted
- **Risk avoidance** – avoid the environment in which the risk occurs e.g. buying an OTS application would avoid a lot of the risks associated with software development e.g. poor estimates of effort.
- **Risk reduction** – the risk is accepted but actions are taken to reduce its likelihood e.g. prototypes ought to reduce the risk of incorrect requirements
- **Risk transfer** – the risk is transferred to another person or organization. The risk of incorrect development estimates can be transferred by negotiating a fixed price contract with an outside software supplier.
- **Risk mitigation** – tries to reduce the impact if the risk does occur e.g. taking backups to allow rapid recovery in the case of data corruption



Risk acceptance

This is the do-nothing option. We will already, in the risk prioritization process, have decided to ignore some risks in order to concentrate on the more likely or damaging. We could decide that the damage inflicted by some risks would be less than the costs of action that might reduce the probability of a risk happening.

Risk avoidance

Some activities may be so prone to accident that it is best to avoid them altogether. If you are worried about sharks then don't go into the water. For example, given all the problems with developing software solutions from scratch, managers might decide to retain existing clerical methods, or to buy an off-the-shelf solution.

Risk reduction

It must be appreciated that each risk reduction action is likely to in-

Here we decide to go ahead with a course of action despite the risks, but take precautions that reduce the probability of the risk.

volve some cost. This is discussed in the next section.

This chapter started with a scenario where two of the staff scheduled to work on Amanda's development project at IOE departed for other jobs. If this has been identified as a risk, steps might have been taken to reduce possible departures of staff. For instance, the developers might have been promised generous bonuses to be paid on successful completion of the project.

Recall that Brigette had a problem at Brightmouth College: after the purchase of the payroll package, a requirement for the payroll database to be accessed by another application was identified. Unfortunately, the application that had been bought did not allow such access. An alternative scenario might have been that Brigette identified this as a possible risk early on in the project. She might have come across Richard Fairley's four COTS (commercial off-the-shelf) software acquisition risks – see Table 7.5 – where one risk is difficulty in integrating the data formats and communication protocols of different applications. Brigette might have specified that the selected package must use a widely accepted data management system like Oracle that allows easier integration.

TABLE 7.5 Fairley's four commercial off-the-shelf (COTS) software acquisition risks

Integration	Difficulties in integrating the data formats and communication protocols of different applications.
Upgrading	When the supplier upgrades the package, the package might no longer meet the users' precise requirements. Sticking with the old version could mean losing the supplier's support for the package.
No source code	If you want to enhance the system, you might not be able to do so as you do not have access to the source code.
Supplier failures or buyouts	The supplier of the application might go out of business or be bought out by a rival supplier.

See R. Fairley (1994) 'Risk management for software projects' *IEEE Software* 11(3) 57–67.

Risk mitigation can sometimes be distinguished from risk reduction. *Risk reduction* attempts to reduce the likelihood of the risk occurring. *Risk mitigation* is action taken to ensure that the impact of the risk is lessened when it occurs. For example, taking regular back-ups of data storage would reduce the impact of data corruption but not its likelihood. Mitigation is closely associated with contingency planning which is discussed presently.

Risk transfer

Risk transfer is what effectively happens when you buy insurance.

In this case the risk is transferred to another person or organization. With software projects, an example of this would be where a software development task is outsourced to an outside agency for a fixed fee. You might expect the supplier to quote a higher figure to cover the risk that the project takes longer than the 'average' expected time. On the other hand, a well-established external organization might have productivity advantages as its developers are experienced in the type of development to be carried out. The need to compete with other software development specialists would also tend to drive prices down.

7.9 Risk Management

Contingency

Risk reduction activities would appear to have only a small impact on reducing the probability of some risks, for example staff absence through illness. While some employers encourage their employees to adopt a healthy lifestyle, it remains likely that some project team members will at some point be brought down by minor illnesses such as flu. These kinds of risk need a *contingency plan*. This is a planned action to be carried out if the particular risk materializes. If a team member involved in urgent work were ill then the project manager might draft in another member of staff to cover that work.

The preventative measures that were discussed under the 'Risk reduction' heading above will usually incur some cost regardless of the risk materializing or not. The cost of a contingency measure will only be incurred if the risk actually materializes. However, there may be some things that have to be done in order for the contingency action to be feasible. An obvious example is that back-ups of a database have to be taken if the contingency action when the database is corrupted is to restore it from back-ups. There would be a cost associated with taking the back-ups.

Exercise 7.4

In the case above where staff could be absent through illness, what preconditions could facilitate contingency actions such as getting other team members to cover on urgent tasks? What factors would you consider in deciding whether these preparatory measures would be worthwhile?

Deciding on the risk actions

Five generic responses to a risk have been discussed above. For each actual risk, however, specific actions have to be planned. In many cases experts have produced lists recommending practical steps to cope with the likelihood of particular risks; see, for example, Boehm's 'top ten' software engineering risks in Table 7.1.

Whatever the countermeasures that are considered, they must be cost-effective. On those occasions where a risk exposure value can be calculated as a financial value using the (*value of damage*) × (*probability of occurrence*) formula – recall Section 7.7 – the cost-effectiveness of a risk reduction action can be assessed by calculating the *risk reduction leverage* (*RRL*).

Risk reduction leverage

Risk reduction leverage = $(RE_{before} - RE_{after})/$ (cost of risk reduction)

RE_{before} is risk exposure before risk reduction e.g. 1% chance of a fire causing £200k damage

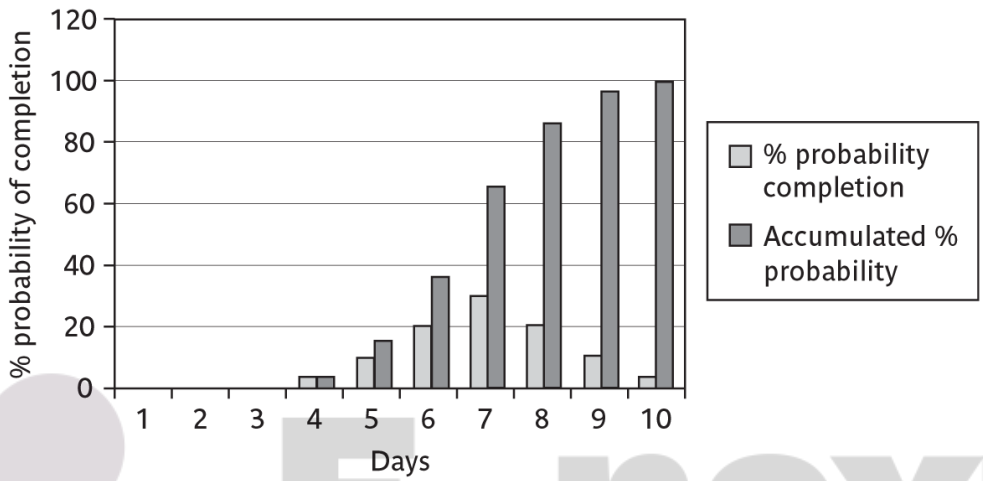
RE_{after} is risk exposure after risk reduction e.g. fire alarm costing £500 reduces probability of fire damage to 0.5%

RRL = (1% of £200k)-(0.5% of £200k)/£500 = 2

RRL > 1.00 therefore worth doing

You could think in terms of the analogy to insurance. An insurance company might reduce the fire insurance premium from £2k to £1k on condition that a fire alarm is installed. The insured would save £1k a year by investing £500 so it would be worth doing.

Probability chart



As was pointed out in Chapter 5, an estimate of activity duration is realistically a range of values clustered around the most likely value are trailing off on either side of this central value.

How can we take account of this in project planning? An answer might be the PERT risk technique.

7.11 Boehm's Top 10 Risks and Counter Measures

Boehm has identified the top 10 risks that a typical project suffers from and has recommended a set of countermeasures for each. We briefly review these in the following.

Barry W. Boehm, 'Software Risk Management: Principles and Practices,' *IEEE Software*, Volume 8, Issue 1, January 1991.

1. **Personnel shortfall:** This risk concerns shortfall of project personnel. The shortfall may show up as either project personnel may lack some specific competence required for the project tasks or personnel leaving the project (called manpower turnover) before project completion. The countermeasures suggested include staffing with top talent, job matching, team building, and cross-training of personnel.
2. **Unrealistic schedules and budgets:** The suggested counter measures include the project manager working out the detailed milestones and making cost and schedule estimations based on it. Other counter measures are incremental development, software reuse, and requirements scrubbing. It may be mentioned that requirements scrubbing involves removing the overly complex and unimportant requirements, in consultation with the customers.
3. **Developing the wrong functions:** The suggested countermeasures include user surveys and user participation, developing prototypes and eliciting user feedback, and early production users' manuals and getting user feedback on it.

4. **Developing the wrong user interface:** The countermeasures suggested for this risk include prototyping, scenarios and task analysis, and user participation.
5. **Gold-plating:** Gold-plating as discussed in Chapter 1, concerns development of features that the team members consider nice to have and, therefore, decide to develop those even though the customer has not expressed any necessity for those. The countermeasures suggested for this risk includes requirements scrubbing, prototyping and cost-benefit analysis.
6. **Continuing stream of requirements changes:** The countermeasures suggested for this risk include incremental development, high change threshold and information hiding.
7. **Shortfalls in externally-furnished components:** This concerns the risk that the components developed by third party are not up to the mark. The countermeasures suggested for this risk include benchmarking, inspections, reference checking and compatibility analysis.
8. **Shortfalls in externally performed tasks**: This concerns the risk that the work performed by the contractors may not be up to the mark. The countermeasures suggested for this risk include reference checking, pre-award audits, award-fee contracts, competitive design or prototyping and team building.
9. **Real-time performance shortfalls:** The countermeasures suggested for this risk include simulation, benchmarking, modelling, prototyping, instrumentation and tuning.
10. **Straining computer science capabilities**: The countermeasures suggested for this risk include technical analysis, cost-benefit analysis and prototyping.

Using PERT to evaluate the effects of uncertainty

Three estimates are produced for each activity

- *Most likely time (m)*
- *Optimistic time (a)*
- *Pessimistic (b)*
- 'expected time' $t_e = (a + 4m + b) / 6$
- 'activity standard deviation' $S = (b-a)/6$

- *Most likely time (m)* the time we would expect the task to take normally
- *Optimistic time (a)* the shortest time that could be realistically be expected
- *Pessimistic (b)* worst possible time (only 1% chance of being worse, say)
- Some straightforward activities (data input of standing data might perhaps be an example) might have little uncertainty and therefore have a low standard deviation, while others (software design, for instance?) have more uncertainty and would have a bigger standard deviation.

A chain of activities

Task	a	m	b	t_e	s
A	10	12	16	?	?
B	8	10	14	?	?
C	20	24	38	?	?

Fill the missing gaps?

Task A t_e = (10+ (12 x 4) + 16)/ 6 i.e. 12.66 s = (16-10)/6 i.e. 1

Task B t_e = (8 + (10 x 4) + 14)/ 6 i.e. 10.33 s = (14-8)/6 i.e. 1

Task C T_e = (20 + (24 x 4) + 38)/6 i.e. 25.66 s = (38-20)/6 i.e. 3

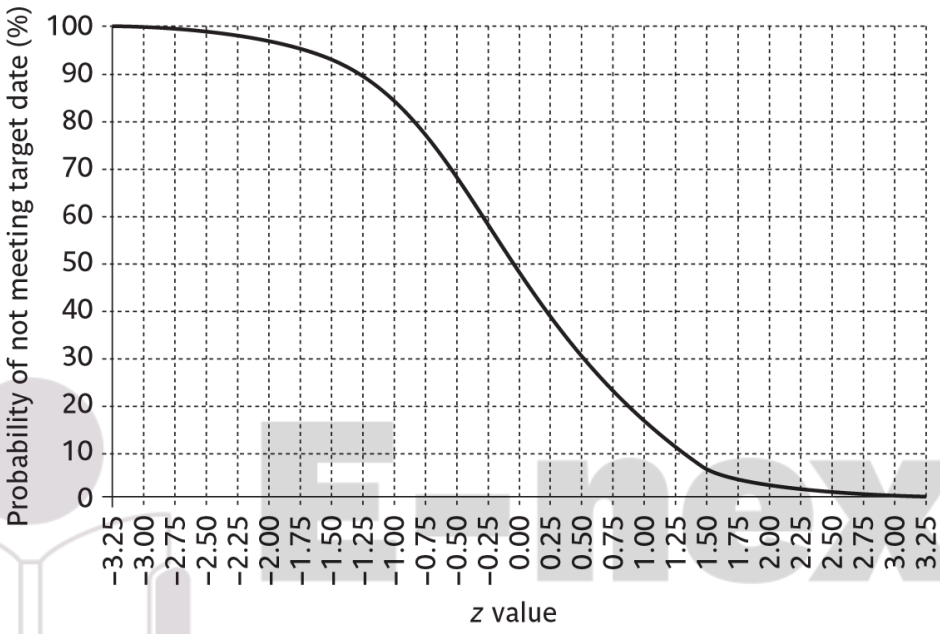
- What would be the expected duration of the chain A + B + C?
- Answer: 12.66 + 10.33 + 25.66 i.e. 48.65
- What would be the standard deviation for A + B+ C?
- Answer: square root of (1^2 + 1^2 + 3^2) i.e. 3.32

Assessing the likelihood of meeting a target

- Say the target for completing A+B+C was 52 days (T)
- Calculate the z value thus
  $z = (T - t_e)/s$
- In this example z = (52-48.33)/3.32 i.e. 1.01
- Look up in table of z values – see next overhead

The target, in this case 52 days, is one that could be based on an externally imposed deadline e.g. the start of the London Olympics. The STANDARDIZE calculation in Excel can be used to calculate the Z value.

Graph of z values



There is about a 15% chance of not meeting the target of 52 days. The Excel NORMSDIST can be used to do this calculation.

Monte Carlo Simulation

- An alternative to PERT.
- A class of general analysis techniques:
  - Valuable to solve any problem that is complex, nonlinear, or involves more than just a couple of uncertain parameters.
- Monte Carlo simulations involve repeated random sampling to compute the results.
- Gives more realistic results as compared to manual approaches.

Steps of a Monte Carlo Analysis

1. Assess the range for the variables being considered.
2. Determine the probability distribution of each variable.
3. For each variable, select a random value based on the probability distribution.
4. Run a deterministic analysis or one pass through the model.
5. Repeat steps 3 and 4 many times to obtain the probability distribution of the model's results.

Critical chain concept

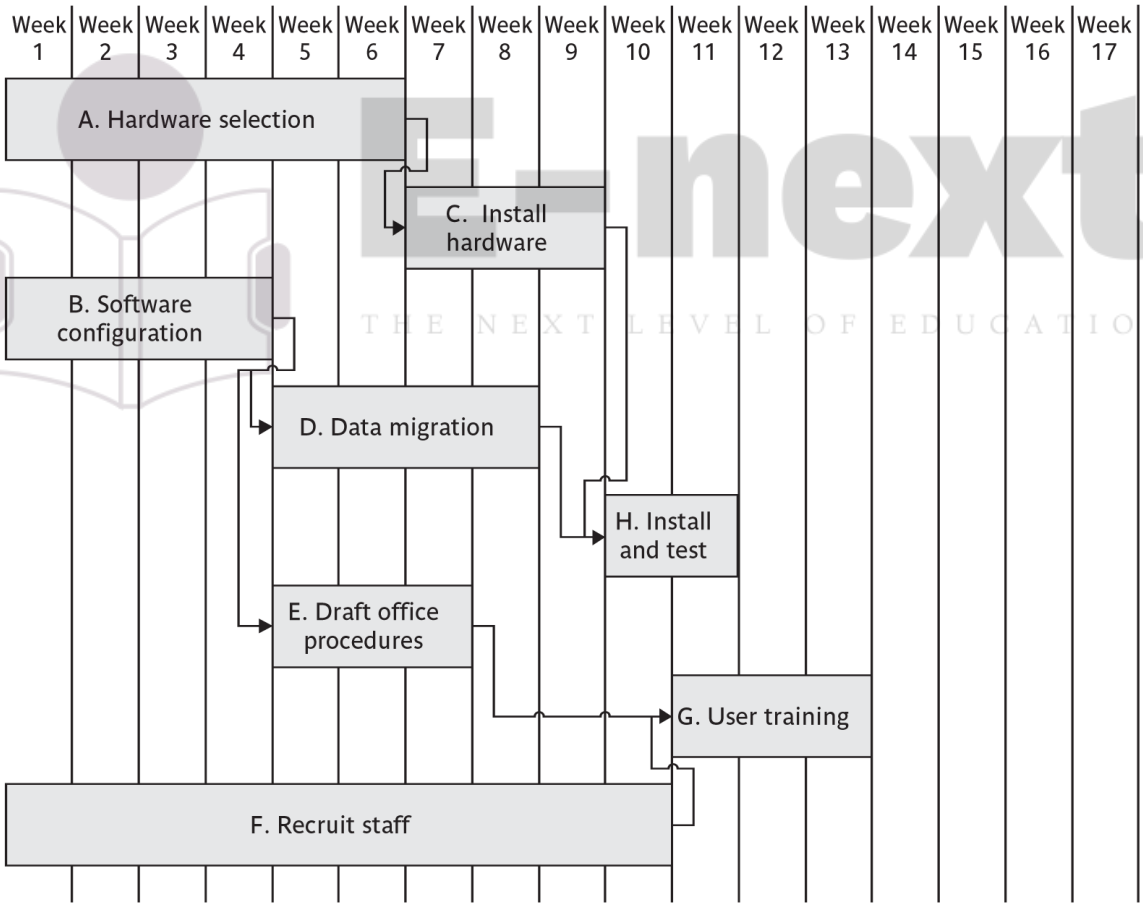


Fig. Traditional planning approach

Note that this plan tends to start activities as early as possible – for example see Activities E and H. While the float on these activities could be used as a buffer if they were late, other activities e.g. F. although of substantial duration do not have the same kind of cushion. In the case of F, the planner would have to make any safety buffer part of the core activity duration.

Critical chain approach

One problem with estimates of task duration:

- Estimators add a safety zone to estimate to take account of possible difficulties
- Developers work to the estimate + safety zone, so time is lost
- No advantage is taken of opportunities where tasks can finish early – and provide a buffer for later activities

Developers will tend to start activities as late as is compatible with meeting the target date as they often have other urgent work to be getting on with in the mean time.

One answer to this:

1. Ask the estimators for two estimates
   - Most likely duration: 50% chance of meeting this
   - Comfort zone: additional time needed to have 95% chance
2. Schedule all activities using most likely values and starting all activities on latest start dates

This approach means that the 'safety buffer' in the estimate for an activity is moved from the individual developer to the project as a whole.

Using latest start dates for activities

Working backwards from the target completion date, each activity is scheduled to start as late as possible. Among other things, this should reduce the chance of staff being pulled off the project on to other work. It is also argued – with some justification according to van Genuchten's research above – that most developers would tend to start work on the task at the latest start time anyway. However, it does make every activity 'critical'. If one is late the whole project is late. That is why the next steps are needed.

Activity	Most likely	Plus comfort zone	Comfort zone
A	6	8	2
B	4	5	1
C	3	3	0
D	4	5	1
E	3	4	1
F	10	15	5
G	3	4	1
H	2	2.5	0.5

TABLE 7.8 Most likely and comfort zone estimates (days)

3. Identify the critical chain – same a critical path but resource constraints also taken into account
4. Put a project buffer at the end of the critical chain with duration 50% of sum of comfort zones of the activities on the critical chain.
5. Where subsidiary chains of activities feed into critical chain, add feeding buffer
6. Duration of feeding buffer 50% of sum of comfort zones of activities in the feeding chain
7. Where there are parallel chains, take the longest and sum those activities

Worked example

Figure 7.12 shows the results of this process. The critical chain in this example happens to be the same as the critical path, that is activities F and G which have comfort zones of 5 weeks and 1 week respectively, making a total of 6 weeks. The project buffer is therefore 3 weeks.

FIGURE 7.12 Gantt chart – critical chain planning approach

Subsidiary chains feed into the critical chain where activity H links into the project buffer and where activity E links into G which is part of the critical chain. Feeding buffers are inserted at these points. For the first buffer the duration would be 50% of the saved comfort zones of A, C and H, that is $(2 + 0 + 0.5)/2 = 1.25$ weeks. It could be argued that B, D and H could form a feeder chain which also has a combined comfort zone of $(1 + 1 + 0.5)/2 = 1.25$ weeks. In the situations where there are parallel alternative paths on a feeder chain, the practice is to base the feeding buffer on whichever comfort zone total is greater. This because if one or other or both parallel paths were late they could still use the same buffer. (Imagine that in the example

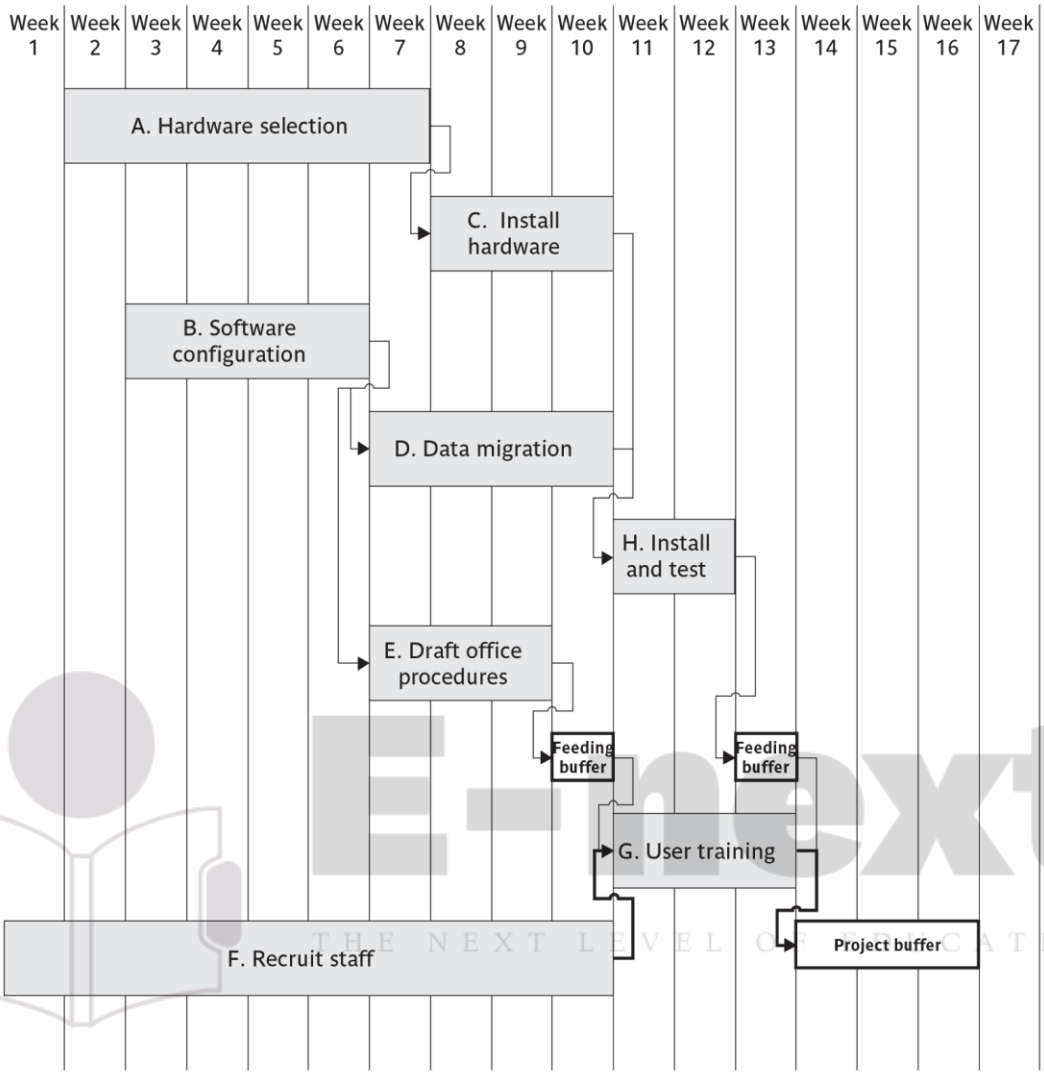


Fig:7.12 Plan employing critical chain concepts

meeting – they might each add a 15-minute comfort zone to the ride on that day but that 15 minutes could effectively be the same 15 minutes between 7.45 and 8.00 a.m. in the morning). It could be argued that the feeding buffer and the final project buffer could also be merged, but explanations of critical chain planning, such as that of Larry Leach (see above), make clear that this is not to be done. This could be because a delay penetrating the feeding buffer time does not affect the completion date of the project, while penetrating the project buffer does.

In the second place, where a feeder chain of activities joins the critical chain, the feeder buffer would be 50% of the comfort zones of activities B and E, that is 1 week.

Executing the critical chain-based plan

- No **chain** of tasks is started earlier than scheduled, but once it has started is finished as soon as possible
- This means the activity following the current one starts as soon as the current one is completed, even if this is early – the relay race principle

Buffers are divided into three zones:

- **Green:** the first 33%. No action required
- **Amber :** the next 33%. Plan is formulated
- **Red :** last 33%. Plan is executed.

Project execution

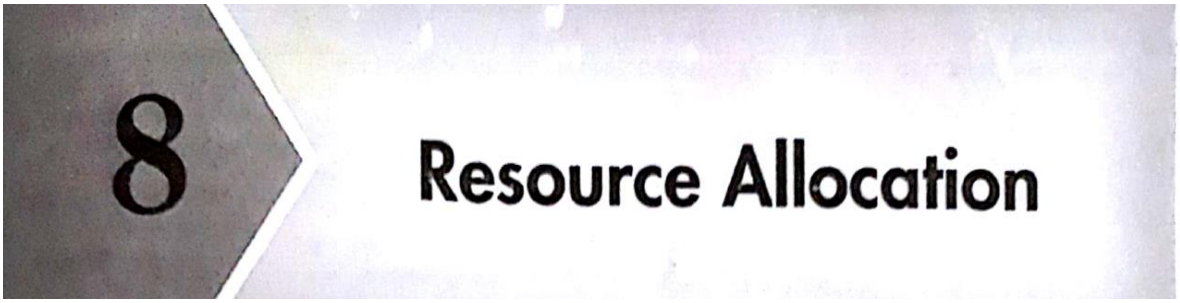
When the project is executed, the following principles are followed:

- No chain of tasks should be started earlier than scheduled, but once it has been started it should be finished as soon as possible – this invokes the *relay race principle*, where developers should be ready to start their tasks as soon as the previous, dependent, tasks are completed.
- Buffers are divided into three zones: green, amber and red, each of an even (33%) size:
  - *green*, where no action is required if the project completion date creeps into this zone;
  - *amber*, where an action plan is formulated if the project completion dates moves into this zone;
  - *red*, where the action plan above is executed if the project penetrates this zone.

Critical chain planning concepts have the support of a dedicated group of enthusiasts. However, the full application of the model has attracted controversy on various grounds. Our personal view is that the ideas of:

- requiring two estimates: the most likely duration/effort and the safety estimate which includes additional time to deal with problems that could arise with the task, and
- placing the contingency time, based on the 'comfort zone' which is the difference between the most likely and safety estimates, in common buffers rather than associating it with individual activities are sound ones that could usefully be absorbed into software project management practice.

See, for example, D. Trietsch (2005) 'Why a critical path by any other name would smell less sweet' *Project Management Journal* 36(1) 27–36 and T. Raz *et al.* (2003) 'A critical look at critical chain project management' *Project Management Journal* 34(4) 24–32.



Schedules

- ***Activity schedule*** - indicating start and completion dates for each activity
- ***Resource schedule*** - indicating dates when resources needed + level of resources
- ***Cost schedule*** showing accumulative expenditure

Resources

- These include
  - labour
  - equipment (e.g. workstations)
  - materials
  - space
  - services
- Time: elapsed time can often be reduced by adding more staff
- Money: used to buy the other resources

8.2 Nature of Resources

A resource is any item or person required for the execution of the project. This covers many things – from paper clips to key personnel – and it is unlikely that we would wish to itemize every resource required, let alone draw up a schedule for their use. Stationery and other standard office supplies, for example, need

not normally be the concern of the project manager – ensuring an adequate supply is the role of the office manager. The project manager must concentrate on those resources which, without planning, might not be available when required.

Some resources, such as a project manager, will be required for the duration of the project whereas others, such as a specific software developer, might be required for a single activity. The former, while vital to the success of the project, does not require the same level of scheduling as the latter. As we saw in Chapter 2, a project manager may not have unrestricted control over a developer who may be needed to work on a range of projects. The manager may have to request the use of a developer who belongs to a pool of resources controlled at programme level.

In general, resources will fall into one of seven categories.

- *Labour* The main items in this category will be members of the development project team such as the project manager, systems analysts and software developers. Equally important will be the quality assurance team and other support staff and any employees of the client organization who might be required to undertake or participate in specific activities.
- *Equipment* Obvious items will include workstations and other computing and office equipment. We must not forget that staff also need basic equipment such as desks and chairs.
- *Materials* Materials are items that are consumed, rather than equipment that is used. They are of little consequence in most software projects but can be important for some – software that is to be widely distributed might, for example, require supplies of disks to be specially obtained.
- *Space* For projects that are undertaken with existing staff, space is normally readily available. If any additional staff (recruited or contracted) should be needed then office space will need to be found.
- *Services* Some projects will require procurement of specialist services – development of a wide area distributed system, for example, requires scheduling of telecommunications services.
- *Time* Time is the resource that is being offset against the other primary resources – project timescales can sometimes be reduced by increasing other resources and will almost certainly be extended if they are unexpectedly reduced.
- *Money* Money is a secondary resource – it is used to buy other resources and will be consumed as other resources are used. It is similar to other resources in that it is available at a cost – in this case interest charges.

The cost of money as a resource is a factor taken into account in DCF evaluation.

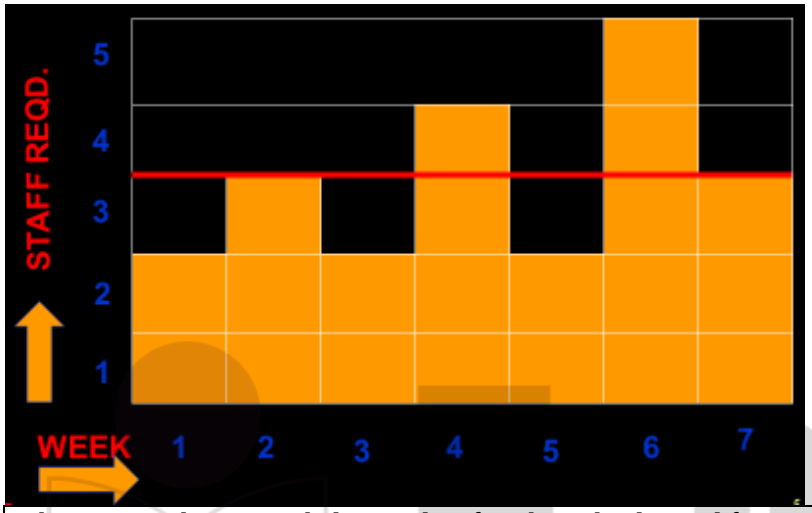
Resource allocation

- Identify the resources needed for each activity and create a ***resource requirement list***
- Identify ***resource types*** - individuals are interchangeable within the group (e.g. 'VB programmers' as opposed to 'software developers')
- Allocate resource types to activities and examine the ***resource histogram***

- This is covered in Section 8.3 of the text.
- Note that at this point we have to assume that we are dealing with, for example, 'standard' software developers who have an average productivity. When we allocate actual people we may find that we have a trainee or a super-expert and this will affect productivity. A shortcoming in productivity in an individual might be compensated for by a lower cost (as would be expected with trainees).

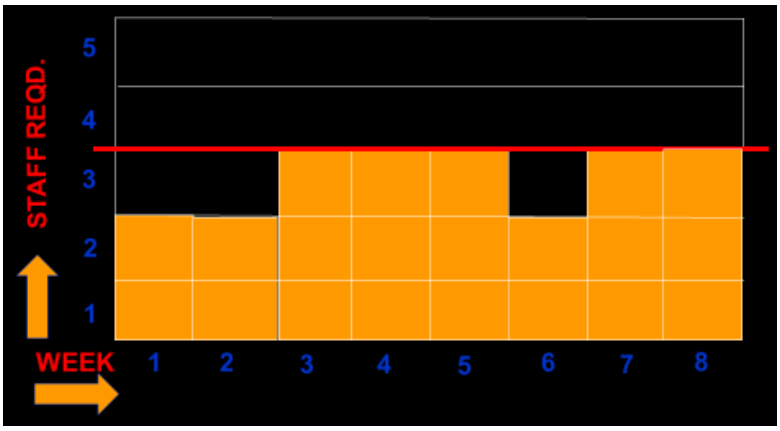
- In the example in the text we start by scheduling every activity to start at the earliest possible date. However in Chapter 7, in the section on the critical chain technique it was suggested that we plan to start activities as late as possible. Whatever the starting procedure, we then need to deal with resource clashes.

Resource histogram: systems analysts



The resource histogram helps us identify where the demand for a resource exceeds the supply.

If we use a tool such as MS Project, the tool will generate the resource histograms for us.



Resource smoothing

- It is usually difficult to get specialist staff who will work odd days to fill in gaps – need for staff to learn about application etc
- Staff often have to be employed for a continuous block of time
- Therefore desirable to employ a constant number of staff on a project – who as far as possible are fully employed
- Hence need for **resource smoothing**

Changing the level of resources on a project over time, particularly personnel, generally adds to the cost of a project. Recruiting staff has costs and, even where staff are transferred internally, time will be needed for familiarization with the new project environment.

The resource histogram in Figure 8.3 poses particular problems in that it calls for two analyst/designers to be idle for twelve days, one for seven days and one for two days between the specification and design stage. It is unlikely that IOE would have another project requiring their skills for exactly those periods of time. This raises the question whether this idle time should be charged to Amanda's project. The ideal resource histogram will be smooth with, perhaps, an initial build-up and a staged run-down.

An additional problem with an uneven resource histogram is that it is more likely to call for levels of resource beyond those available. Figure 8.4 illustrates how, by adjusting the start date of some activities and splitting others, a resource histogram can, subject to constraints such as precedence requirements, be smoothed to contain resource demand at available levels. The different letters represent staff working on a series of module testing tasks, that is, one person working on task A, two on tasks B and C etc.

In Figure 8.4, the original histogram was created by scheduling the activities at their earliest start dates. The resource histogram shows the typical peaked shape caused by earliest start date scheduling and calls for a total of nine staff where only five are available for the project.

By delaying the start of some of the activities, it has been possible to smooth the histogram and reduce the maximum level of demand for the resource. Notice that some activities, such as C and D, have been split. Where non-critical activities can be split they can provide a useful way of filling troughs in the demand for a resource, but in software projects it is difficult to split tasks without increasing the time they take.

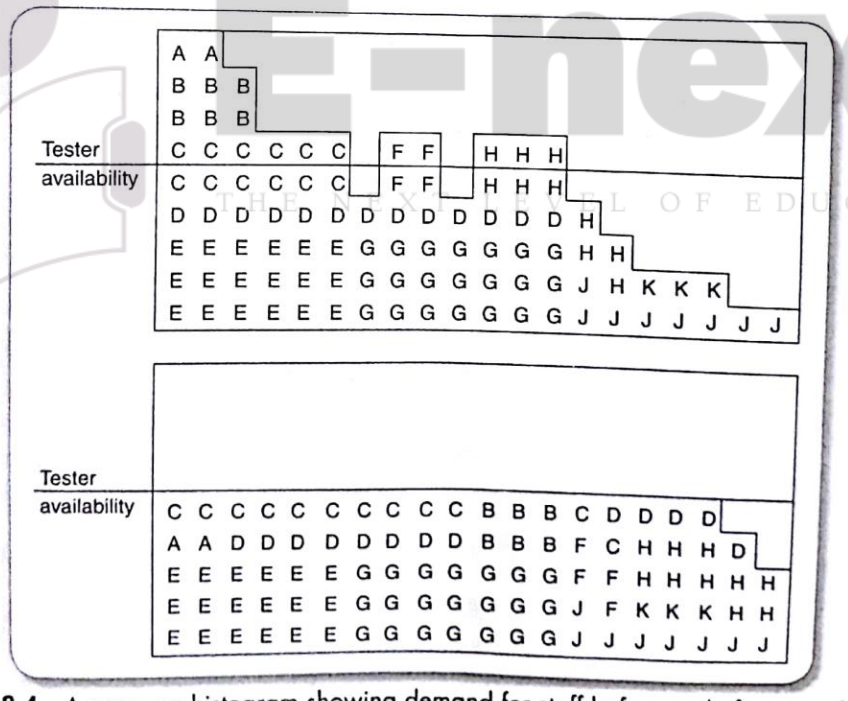


FIGURE 8.4 A resource histogram showing demand for staff before and after smoothing

Resource clashes

- Where same resource needed in more than one place at the same time
- can be resolved by:

- delaying one of the activities
  - taking advantage of float to change start date
  - delaying start of one activity until finish of the other activity that resource is being used on - *puts back project completion*
- moving resource from a non-critical activity
- bringing in additional resource - *increases costs*

Prioritizing activities

There are two main ways of doing this:

- ***Total float priority*** – those with the smallest float have the highest priority
- ***Ordered list priority*** – this takes account of the duration of the activity as well as the float – see next overhead

Where more than one activity is competing for the same limited resource at the same time then those activities need to be prioritized.

Burman's priority list

Give priority to:

- Shortest critical activities
- Other critical activities
- Shortest non-critical activities
- Non-critical activities with least float
- Non-critical activities

Resource usage

- Need to maximise %usage of resources i.e. reduce idle periods between tasks
- Need to balance costs against early completion date
- Need to allow for contingency

Critical path

- Scheduling resources can create new dependencies between activities – recall *critical chains*
- It is best not to add dependencies to the activity network to reflect resource constraints
  - Makes network very messy
  - A resource constraint may disappear during the project, but link remains on network
- Amend dates on **schedule** to reflect resource constraints

- In chapter 7 the concept of critical chains was introduced which took account of resource constraints.
- The point about not adding links to the network to deal with resource constraints is not in the text, but is based on practical experience. The notation for activity networks does not tell you why one activity might be dependent on the completion of another.

Allocating individuals to activities

The initial 'resource types' for a task have to be replaced by actual individuals.

Factors to be considered:

- Availability
- Criticality
- Risk
- Training
- Team building – and motivation

- **Availability** – who is free? Note that this will change during the course of the project as some tasks are completed earlier or later than planned
- **Criticality –** You would want to put your more experienced, 'safer', staff on the critical activities
- **Risk –** this is similar to the point above, but some activities could be off the critical path but still have risks e.g. to the quality of subsequent products
- **Training** – despite concerns about minimizing risk, it is healthy to take some risks in order to develop staff capabilities by allocating challenging tasks to relatively inexperienced stafff

- **Team-building** – identifying people who work well together can pay dividends; chopping and changing plans all the time may in theory optimize project performance, but can in practice be demotivating for staff

Cost schedules

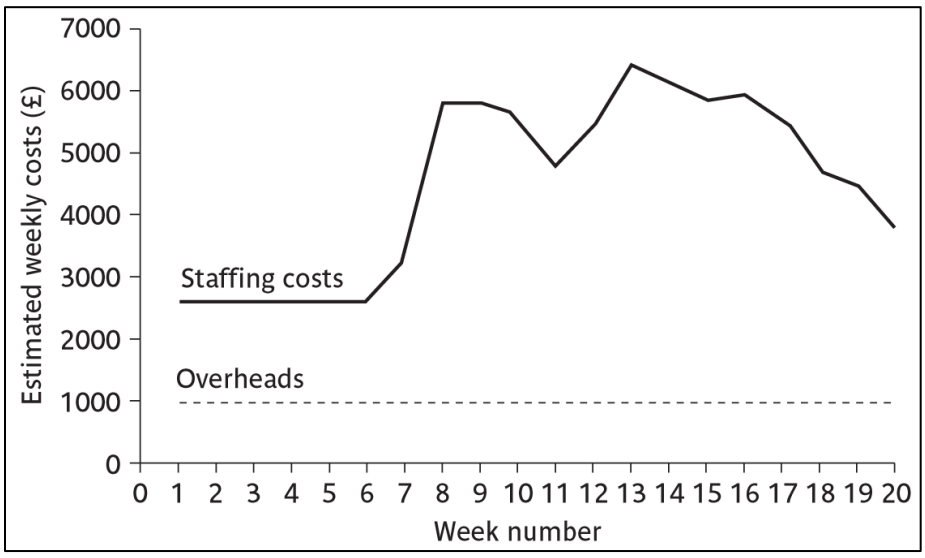
Cost schedules can now be produced:

Costs include:

- Staff costs
- Overheads
- Usage charges

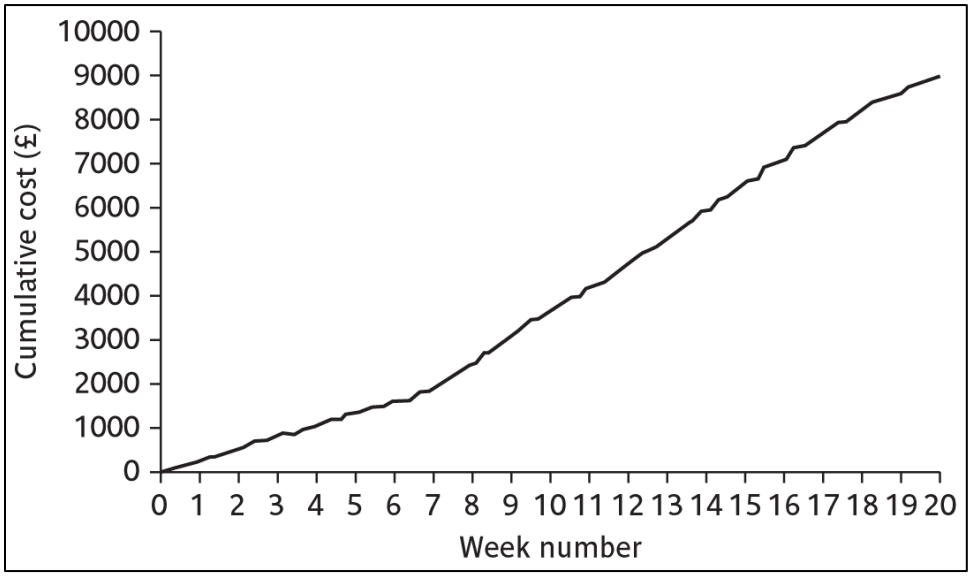
- Section 8.9 covers this in more depth.
- Staff costs – includes not just salary, but also social security contributions by the employer, holiday pay etc. Timesheets are often used to record actual hours spent on each project by an individual. One issue can be how time when a staff member is allocated and available to the project, but is not actually working on the project, is dealt with.
- Overheads e.g. space rental, service charges etc. Some overheads might be directly attributable to the project; in other cases a percentage of departmental overheads may be allocated to project costs.
- Usage charges – some charges can be on a 'pay as you go' basis e.g. telephone charges, postage, car mileage – at the planning stage an estimate of these may have to be made

Cost profile



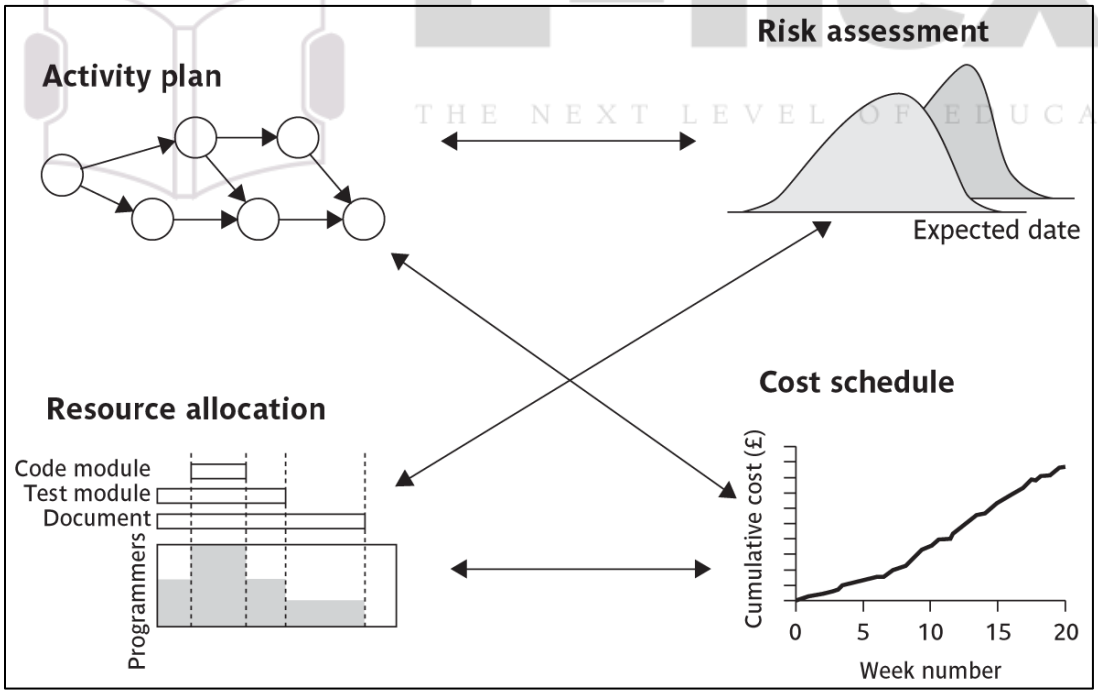
This shows how much is going to be spent in each week. This could be important where an organization allocates project budgets by financial year or quarter and the project straddles more than one of these financial periods

Accumulative costs



The project manager will also be concerned about planned accumulative costs. This chart can be compared to the actual accumulative costs when controlling the project to assess whether the project is likely to meet its cost targets.

Balancing concerns



Successful project scheduling is not a simple sequence.

Because of the inter-linking of different concerns project planning will need to be iterative. The consequences of decisions will need to carefully assessed and plans adjusted accordingly.