Chapter 1

Introduction to software project management

OBJECTIVES

When you have completed this chapter you will be able to:

- define the scope of 'software project management';
- distinguish between software and other types of development project;
- understand some problems and concerns of software project managers;
- define the usual stages of a software project;
- explain the main elements of the role of management;
- understand the need for careful planning, monitoring and control;
- identify the stakeholders of a project, their objectives and ways of measuring the success in meeting those objectives;
- measure the success of a project in meeting its objectives.

1.1 Introduction

What exactly do we mean by 'software project management'? To answer this, we need to look at some key ideas about the planning, monitoring and control of software projects. Projects to produce software are worthwhile only if they satisfy real needs and so we will examine how we can identify the stakeholders in a project and their objectives. Identifying those objectives and checking that they are met is the basis of a successful project. This, however, cannot be done unless there is accurate information and how this is provided will be explored.

1.2 What is a project?

Dictionary definitions of 'project' include:
'A specific plan or design'
'A planned undertaking'
'A large undertaking: for example, a public works scheme'
Longman Concise English Dictionary, 1982.

The dictionary definitions put a clear emphasis on the project's being a planned activity.

Another key aspect of a project is that the undertaking is non-routine: a job which is repeated a number of times is not a project. There is a hazy boundary in between. The first time you do a routine task it will be very like a project. On the other hand, a project to develop a system that is very similar to previous ones that you have developed will have a large element of the routine.

We can summarize the key characteristics that distinguish projects as follows:

- non-routine tasks are involved;
- planning is required;
- specific objectives are to be met or a specified product is to be created;
- the project has a predetermined time span (which may be absolute or relative);
- work is carried out for someone other than yourself;
- work involves several specialisms;
- work is carried out in several phases;
- the resources that are available for use on the project are constrained;
- the project is large or complex.

In general, the more any of these factors applies to a task, the more difficult it is going to be to complete it successfully. Project size is particularly important. It should not be a surprise that a project that employs 200 project personnel is going to be rather trickier to manage than one that involves just two people. The examples and exercises used in this book usually relate to smaller projects. This is just to make them more manageable from a learning point of view: the techniques and issues discussed are of equal relevance to larger projects.

Exercise 1.1

Consider the following:

- producing an edition of a newspaper;
- building the Channel Tunnel;
- getting married;
- amending a financial computer system to deal with dates after the 31st December, 1999;
- a research project into what makes a good human–computer interface;
- an investigation into the reason why a user has a problem with a computer system;

- a programming assignment for a second year computing student;
- writing an operating system for a new computer;
- installing a new version of a word processing application in an organization.

Some would appear to merit the description 'project' more than others. Put them into an order that most closely matches your ideas of what constitutes a project. For each entry in the ordered list, describe the difference between it and the one above that makes it less worthy of the term 'project'.

There is no one correct answer to this exercise, but a possible solution to this and the other exercises you will come across may be found at the end of the book.

1.3 Software projects versus other types of project

Many of the techniques of general project management are applicable to software project management, but Fred Brooks pointed out that the products of software projects have certain characteristics that make them different.

Brooks, F. P. 'No silver bullet: essence and accidents of software engineering'. This essay has been included in *The Mythical Man-Month*, Anniversary Edition, Addison-Wesley, 1995.

One way of perceiving software project management is as the process of making visible that which is invisible.

Invisibility When a physical artefact such as a bridge or road is being constructed the progress being made can actually be seen. With software, progress is not immediately visible.

Complexity Per dollar, pound or euro spent, software products contain more complexity than other engineered artefacts.

Flexibility The ease with which software can be changed is usually seen as one of its strengths. However this means that where the software system interfaces with a physical or organizational system, it is expected that, where necessary, the software will change to accommodate the other components rather than vice versa. This means the software systems are likely to be subject to a high degree of change.

1.4 Activities covered by software project management

A software project is concerned not only with the actual writing of software. In fact, where a software application is bought in 'off-the-shelf', there might be no software writing as such. This is still fundamentally a software project because so many of the other elements associated with this type of project are present.

Chapter 4 on project analysis and technical planning looks at some alternative life cycles.

Usually, there are three successive processes that bring a new system into being:

1. **The feasibility study** This is an investigation to decide whether a prospective project is worth starting. Information will be gathered about the general requirements of the proposed system. The probable developmental

and operational costs, along with the value of the benefits of the new system are estimated. With a large system, the feasibility study could be treated as a project in its own right. This evaluation may be done as part of a strategic planning exercise where a whole range of potential software developments are evaluated and put into an order of priority. Sometimes an organization has a policy where a series of projects is planned as a *programme* of development.

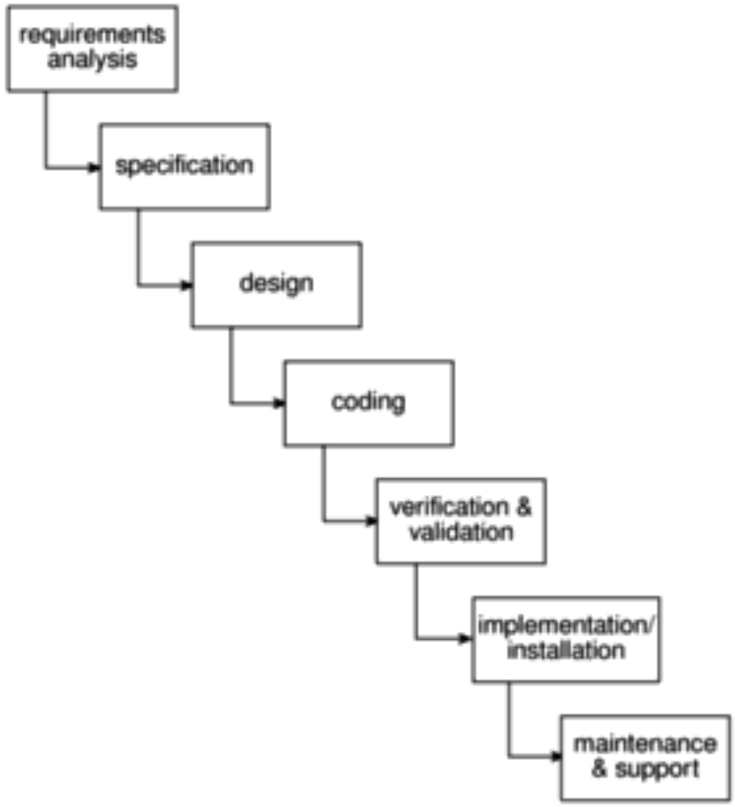


Figure 1.1 *A typical project life-cycle.*

The PRINCE 2 method, which is described in Appendix A takes this planning by stages approach.

2. **Planning** If the feasibility study produces results that indicate that the prospective project appears viable, then planning of the project can take place. In fact, for a large project, we would not do all our detailed planning right at the beginning. We would formulate an outline plan for the whole project and a detailed one for the first stage. More detailed planning of the later stages would be done as they approached. This is because we would have more detailed and accurate information upon which to base our plans nearer to the start of the later stages.

3. **Project execution** The project can now be executed. Individual projects are likely to differ considerably but a classic project life-cycle is shown in Figure 1.1.

The stages in the life-cycle illustrated in Figure 1.1 above are described in a little more detail below:

Requirements analysis This is finding out in detail what the users require of the system that the project is to implement. Some work along these lines will almost

certainly have been carried out when the project was evaluated but now the original information obtained needs to be updated and supplemented. Several different approaches to the users' requirements may be explored. For example, a small system that satisfies some, but not all, of the users' needs at a low price may be compared to a system with more functions but at a higher price.

Specification Detailed documentation of what the proposed system is to do.

Design A design that meets the specification has to be drawn up. This design activity will be in two stages. One will be the external or user design. This lays down what the system is to look like to the users in terms of menus, screen and report layouts and so on. The next stage produces the physical design, which tackles the way in which the data and software procedures are be structured internally.

Coding This might refer to writing code in a procedural language such as C or Ada, or might refer to the use of a high level application builder. Even where software is not being built from scratch, some modification to the base application might be required to meet the needs of the new application.

Verification and validation Whether software is developed specially for the current application or not, careful testing will be needed to check that the proposed system meets its requirements.

Implementation/installation Some system development practitioners refer to the whole of the project after design as 'implementation' (that is, the implementation of the design) while others insist that the term refers to the installation of the system after the software has been developed. In this case it encompasses such things as setting up data files and system parameters, writing user manuals and training users of the new system.

Maintenance and support Once the system has been implemented there will be a continuing need for the correction of any errors that may have crept into the system and for extensions and improvements to the system. Maintenance and support activities may be seen as a series of minor software projects. In many environments, most software development is in fact maintenance.

Exercise 1.2

Brightmouth College is a higher education institution which used to be managed by the local government authority but has now become autonomous. Its payroll is still administered by the local authority and pay slips and other output are produced in the local authority's computer centre. The authority now charges the college for this service. The college management are of the opinion that it would be cheaper to obtain an 'off-the-shelf' payroll application and do the payroll processing themselves.

What would be the main stages of the project to convert to independent payroll processing by the college? Bearing in mind that an off-the-shelf application is to

be used, how would this project differ from one where the software was to be written from scratch?

1.5 Some ways of categorizing software projects

It is important to distinguish between the main types of software project because what is appropriate in one context might not be so in another. For example, SSADM, the Structured Systems Analysis and Design Method, is suitable for developing information systems but not necessarily other types of system.

Information systems versus embedded systems

Embedded systems are also called real-time or industrial systems.

A distinction may be made between *information systems* and *embedded systems*. Very crudely, the difference is that in the former case the system interfaces with the organization, whereas in the latter case the system interfaces with a machine! A stock control system would be an information system that controls when the organization reorders stock. An embedded, or process control, system might control the air conditioning equipment in a building. Some systems may have elements of both so that the stock control system might also control an automated warehouse.

Exercise 1.3

Would an operating system on a computer be an information system or an embedded system?

Objectives versus products

Projects may be distinguished by whether their aim is to produce a *product* or to meet certain *objectives*.

A project might be to create a product the details of which have been specified by the client. The client has the responsibility for justifying the product.

Service level agreements are becoming increasingly important as organizations contract out functions to external service suppliers.

On the other hand, the project might be required to meet certain objectives. There might be several ways of achieving these objectives in contrast to the constraints of the product-driven project. One example of this is where a new information system is implemented to improve some service to users inside or outside an organization. The subject of an agreement would be the level of service rather than the characteristics of a particular information system.

Many software projects have two stages. The first stage is an objectives-driven project, which results in a recommended course of action and may even specify a new software application to meet identified requirements. The next stage is a project actually to create the software product.

Exercise 1.4

Would the project to implement an independent payroll system at the Brightmouth College described in Exercise 1.2 above be an objectives-driven project or a product-driven project?

1.6 The project as a system

A project is concerned with creating a new system and/or transforming an old one and is itself a system.

Systems, subsystems and environments

A simple definition of the term *system* is 'a set of interrelated parts'. A system will normally be part of a larger system and will itself comprise *subsystems*.

Outside the system there will be the system's *environment*. This will be made up of things that can affect the system but over which the system has no direct control. In the case of Brightmouth College, the bankruptcy of the main supplier of IT equipment would be an event happening in the system's environment.

Exercise 1.5

Identify the possible subsystems of the installed Brightmouth College payroll system.

What important entities exist in the payroll system's environment?

Open versus closed systems

Open systems are those that interact with the environment. Nearly all systems are open. One reason that engineered systems and the projects to construct them often fail is that the technical staff involved do not appreciate the extent to which systems are open and are liable to be affected by outside changes.

Sub-optimization

This is where a subsystem is working at its optimum but is having a detrimental effect on the overall system. An example of this might be where software developers deliver to the users a system that is very efficient in its use of machine resources, but is also very difficult to modify.

Sociotechnical systems

Software projects belong to this category of systems. Any software project requires both technological organization and also the organization of people. Software project managers therefore need to have both technical competence and the ability to interact persuasively with other people.

1.7 What is management?

The Open University suggest that management involves the following activities:

A convenient way of accessing this OU material is in D. Ince, H. Sharp, and M. Woodman, *Introduction to Software Project Management and Quality Assurance*, McGraw-Hill, 1993.

- planning – deciding what is to be done;
- organizing – making arrangements;
- staffing – selecting the right people for the job, for example;
- directing – giving instructions;
- monitoring – checking on progress;
- controlling – taking action to remedy hold-ups;
- innovating – coming up with new solutions;
- representing – liaising with users etc.

Exercise 1.6

Paul Duggan is the manager of a software development section. On Tuesday at 10.00 am he and his fellow section heads have a meeting with their group manger about the staffing requirements for the coming year. Paul has already drafted a document 'bidding' for staff. This is based on the work planned for his section for the next year. The document is discussed at the meeting. At 2.00 pm Paul has a meeting with his senior staff about an important project his section is undertaking. One of the software development staff has just had a road accident and will be in hospital for some time. It is decided that the project can be kept on schedule by transferring another team member from less urgent work to this project. A temporary replacement is to be brought in to do the less urgent work but this might take a week or so to arrange. Paul has to phone both the personnel manager about getting a replacement and the user for whom the less urgent work is being done explaining why it is likely to be delayed.

Identify which of the eight management responsibilities listed above Paul was responding to at different points during his day.

Another way of looking at the management task is to ask managers what their most frequent challenges are. A survey of software project managers produced the following list:

The results of this survey by H. J. Thamhain and D. L. Wilemon appeared in June 1986 in *Project Management Journal* under the title 'Criteria for controlling software according to plan'.

- coping with deadlines (85%);
- coping with resource constraints (83%);
- communicating effectively among task groups (80%);
- gaining commitment from team members (74%);
- establishing measurable milestones (70%);
- coping with changes (60%);

- working out project plan agreement with their team (57%);
- gaining commitment from management (45%);
- dealing with conflict (42%);
- managing vendors and sub-contractors (38%).

Similar lists appear in the computer trade press, for example in the 27 August 1998 edition of *Computing*.

The percentages relate to the numbers of managers identifying each challenge. A manager could identify more than one.

1.8 Problems with software projects

One way of deciding what ought to be covered in 'software project management' is to consider what problems need to be addressed.

Traditionally, management has been seen as the preserve of a distinct class within the organization. As technology has made the tasks undertaken by an organization more sophisticated, many management tasks seem to have become dispersed throughout the organization: there are management systems rather than managers. Nevertheless, the successful project will normally have one person who is responsible for its success. Such people are likely to be concerned with the key areas that are most likely to prevent success – they are primarily trouble-shooters and their job is likely to be moulded by the problems that confront the project. A survey of managers published by Thayer, Pyster and Wood identified the following commonly experienced problems:

- poor estimates and plans;
- lack of quality standards and measures;
- lack of guidance about making organizational decisions;
- lack of techniques to make progress visible;
- poor role definition – who does what?
- incorrect success criteria.

Further details of the survey can be found in 'Major issues in software engineering project management' in *IEEE Transactions on Software Engineering*, Volume 7, pp 333–342.

The above list looks at the project from the manager's point of view. What about the staff who make up the members of the project team? Below is a list of the problems identified by a number of students on a degree course in Computing and Information Systems who had just completed a year's industrial placement:

- inadequate specification of work;
- management ignorance of IT;
- lack of knowledge of application area;
- lack of standards;
- lack of up-to-date documentation;

- preceding activities not completed on time – including late delivery of equipment;
- lack of communication between users and technicians;
- lack of communication leading to duplication of work;
- lack of commitment – especially when a project is tied to one person who then moves;
- narrow scope of technical expertise;
- changing statutory requirements;
- changing software environment;
- deadline pressure;
- lack of quality control;
- remote management;
- lack of training.

Note how many of the problems identified by the students stemmed from poor communications. Another common problem identified by this and other groups of students is the wide range of IT specialisms – an organization may be made up of lots of individuals or groups who will be expert in one set of software techniques and tools but ignorant of those used by their colleagues. Communication problems are therefore bound to arise.

Stephen Flower's *Software Failure, Management Failure*, Wiley & Sons, 1996, is an interesting survey of failed computer projects

What about the problems faced by the customers of the products of computer projects? Here are some recent stories in the press:

- the United States Internal Revenue System was to abandon its tax system modernization programme after having spent $4 billion;
- the state of California spent $1 billion on its non-functional welfare database system;
- the £339 million United Kingdom air traffic control system was reported as being two years behind schedule;
- a discount stock brokerage company had 50 people working 14 hours or more a day to correct three months of records clerically—the report commented that the new system had been rushed into operation without adequate testing;
- in the United Kingdom, a Home Office immigration service computerization project was reported as having missed two deadlines and was nine months late;
- the Public Accounts Committee of the House of Commons in the United Kingdom blamed software bugs and management errors for £12 million of project costs in relation to an implementation of a Ministry of Agriculture computer system to administer farm subsidies.

Most of the stories above relate to public sector organizations. This may be misleading—private sector organizations tend to conceal their disasters and in any case many of the public projects above were actually being carried out by private sector contractors. Any lingering faith by users in the innate ability of IT people to plan ahead properly will have been removed by consideration of the 'millennium bug', a purely self-inflicted IT problem. On balance it might be a good idea not to survey users about their problems with IT projects!

1.9 Management control

The project control cycle

Management, in general, can be seen as the process of setting objectives for a system and then monitoring the system to see what its true performance is. In Figure 1.2 the 'real world' is shown as being rather formless. Especially in the case of large undertakings, there will be a lot going on about which management should be aware. As an example, take an IT project that is to replace locally held paper-based records with a centrally-organized database. It might be that staff in a large number of offices that are geographically dispersed need training and then need to use the new IT system to set up the back-log of manual records on the new database. It might be that the system cannot be properly operational until the last record has been transferred. It might also be the case that the new system will be successful only if new transactions can be processed within certain time cycles. The managers of the project ought to be asking questions about such things as how effective training has been, how many records have still to be transferred to the new database and transfer rates. This will involve the local managers in *data collection*. Bare details, such as 'location X has processed 2000 documents' will not be very useful to higher management: *data processing* will be needed to transform this raw *data* into useful *information*. This might be in such forms as 'percentage of records processed', 'average documents processed per day per person' and 'estimated completion date'.

In the example above, the project leader might examine the 'estimated completion date' for completing data transfer for each branch and compare this with the overall target date for completion of this phase of the project. In effect they are comparing actual performance with one aspect of the overall project objectives. They might find that one or two branches are not going to complete the transfer of details in time, and would then need to consider what to do (this is represented in Figure 1.2 by the box *making decisions/plans*). One possibility would be to move staff temporarily from one branch to another. If this is done, there is always the danger that while the completion date for the one branch is pulled back to before the overall target date, the date for the branch from which staff are being moved is pushed forward beyond that date. The project manager would need to calculate carefully what the impact would be in moving staff from particular branches. This is *modelling* the consequences of a potential solution.

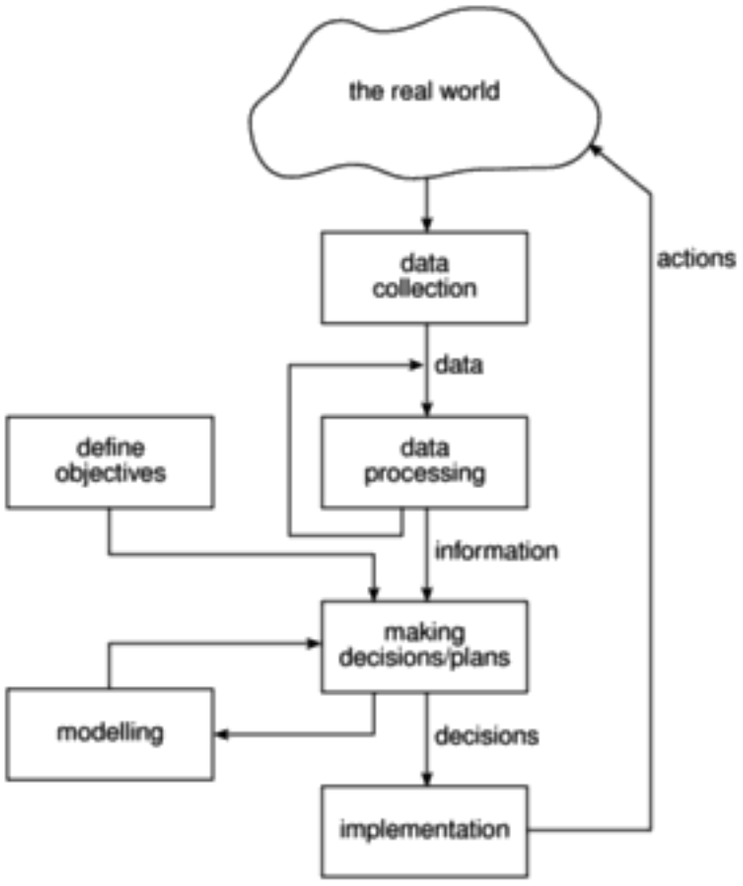


Figure 1.2 *The project control cycle.*

Several different proposals could be modelled in this way before one was chosen for *implementation*.

Having implemented the decision, the situation needs to be kept under review by collecting and processing further progress details. For instance, the next time that progress is reported, a branch to which staff have been transferred might still be behind in transferring details. This might be because the reason why the branch has got behind in transferring details is because the manual records are incomplete and another department, for whom the project has a low priority, has to be involved in providing the missing information. In this case, transferring extra staff to do data input will not have accelerated data transfer.

Objectives

Project objectives should be clearly defined.

To have a successful software project, the manager and the project team members must know what will constitute success. This will make them concentrate on what is essential to project success.

There might be more than one set of users of a system and there might be different groups of staff who are involved its development. There is a need for well defined objectives that are accepted by all these people. Where there is more than one user group, then a *project authority* needs to be identified. Such a project authority has overall authority over what the project is to achieve.

This authority is often held by a *project steering committee*, which has overall responsibility for setting, monitoring and modifying objectives. The project manager still has responsibility for running the project on a day to day basis, but has to report to the steering committee at regular intervals. Only the steering committee can authorize changes to the project objectives and resources.

This committee is likely to contain user, development and management representatives.

Measures of effectiveness

Effective objectives are concrete and well defined. Vague aspirations such as 'to improve customer relations' are unsatisfactory. Objectives should be such that it is obvious to all whether the project has been successful or not. Ideally there should be *measures of effectiveness*, which tell us how successful the project has been. For example, 'to reduce customer complaints by 50%' would be more satisfactory as an objective than 'to improve customer relations'.

The measure can, in some cases, be an answer to simple yes/no question, 'Did we install the new software by 1st June?' for example.

Sub-objectives and goals

In order to keep things manageable, objectives might need to be broken down into sub-objectives. Here we say that in order to achieve A we must achieve B, C and D first. These sub-objectives are also known as *goals*, steps on the way to achieving an objective, just as goals scored in a football match are steps towards the objective of winning the match.

Exercise 1.7

Identify the objectives and sub-objectives of the Brightmouth College payroll project. What measures of effectiveness could be used to check the success in achieving the objectives of the project?

1.10 Stakeholders

These are people who have a stake or interest in the project. It is important that they be identified as early as possible, because you need to set up adequate communication channels with them right from the start. The project leader also has to be aware that not everybody who is involved with a project has the same motivation and objectives. The end users might, for instance, be concerned about the ease of use of the system while their managers might be interested in the staff savings the new system will allow.

Stakeholders might be internal to the project team, external to the project team but in the same organization, or totally external to the organization.

- **Internal to the project team** This means that they will be under the direct managerial control of the project leader.
- **External to the project team but within the same organization** For example, the project leader might need the assistance of the information

management group in order to add some additional data types to a database or the assistance of the users to carry out systems testing. Here the commitment of the people involved has to be negotiated.

- **External to both the project team and the organization** External stakeholders might be customers (or users) who will benefit from the system that the project implements or contractors who will carry out work for the project. One feature of the relationship with these people is that it is likely to be based on a legally binding contract.

Within each of the general categories there will be various groups. For example, there will be different types of user with different types of interests.

B. W. Boehm and R. Ross 'Theory W Software Project Management: Principles and Examples', in B. W. Boehm (ed.) *Software Risk Management.*

Different types of stakeholder might have different objectives and one of the jobs of the successful project leader is to recognize these different interests and to be able to reconcile them. It should therefore come as no surprise that the project leader needs to be a good communicator and negotiator. Boehm and Ross proposed a 'Theory W' of software project management where the manager concentrates on creating situations where all parties involved in a project benefit from it and therefore have an interest in its success. (The 'W' stands for Everyone a Winner.)

Exercise 1.8

Identify the stakeholders in the Brightmouth College payroll project.

1.11 Requirement specification

Very often, especially in the case of product-driven projects, the objectives of the project are carefully defined in terms of functional requirements, quality requirements, and resource requirements.

- **Functional requirements** These define what the system that will be the end product of the project is to do. Systems analysis and design methods, such as SADT and Information Engineering, are designed primarily to provide functional requirements.

These are sometimes called non-functional requirements.

- **Quality requirements** There will be other attributes of the system to be implemented that do not relate so much to what the system is to do but how it is to do it. These are still things that the user will be able to experience. They include, for example, response time, the ease of using the system and its reliability.
- **Resource requirements** A record of how much the organization is willing to spend on the system. There will usually be a trade-off between this and the time it takes to implement the system. In general it costs disproportionately more to implement a system by an earlier date than a later one. There might

also be a trade-off between the functional and quality requirements and cost. We would all like exceptionally reliable and user-friendly systems that do exactly what we want but we might not be able to afford them.

1.12 Information and control in organizations

Hierarchical information and control systems

The larger the project, the bigger the communication problems.

With small projects, the project leaders are likely to be working very closely with the other team members and might even be carrying out many non-managerial tasks themselves. Therefore they should have a pretty good idea of what is going on. When projects are larger, many separate teams will be working on different aspects of the project and the overall managers of the project are not going to have day-to-day direct contact with all aspects of the work.

Larger projects are likely to have a *hierarchical* management structure (Figure 1.3). Project team members will each have a group leader who allocates them work and to whom they report progress. In turn the group leader, along with several other group leaders, will report to a manager at the next higher level. That manager might have to report to another manager at a higher level, and so on.

The referral of disagreements to a higher level is sometimes known as *escalation*.

There might be problems that cannot be resolved at a particular level. For example, additional resources might be needed for some task, or there might be a disagreement with another group. These will be referred to the next higher level of management.

At each higher level more information will be received by fewer people. There is thus a very real danger that managers at the higher levels might be overloaded with too much information. To avoid this, at each level the information will have to be summarized.

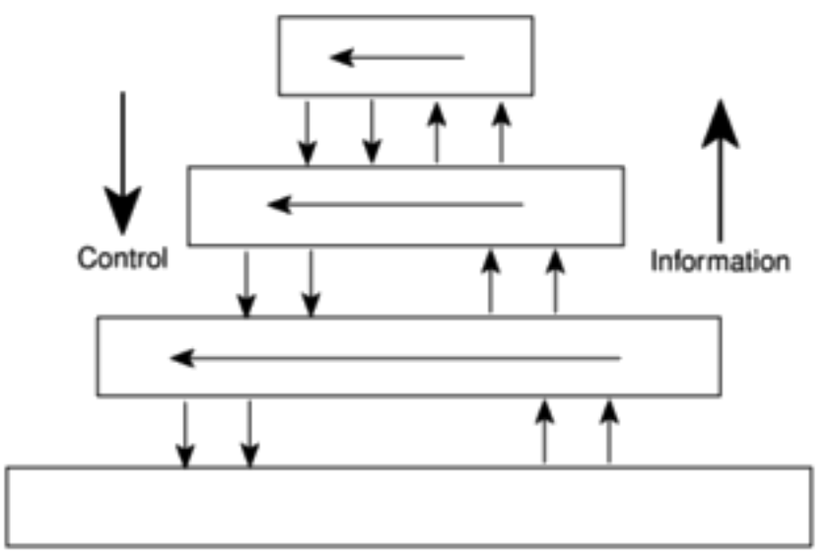


Figure 1.3 *Management information flows up the organizational structure and control flows down.*

As a result of examining the progress information and comparing it against what was planned, some remedial action might need to be taken. Instructions may be formulated and passed down to a lower level of management. The lower level managers will have to interpret what needs to be done and formulate more detailed plans to fulfil the directive. As the directives filter down the hierarchy, they will be expanded into more detail at each level.

For example, a programmer will receive a specification from an analyst and might then seek clarification.

Not all information flows concerning a project will be going up and down the hierarchy. There will also be lateral flows between groups and individuals on the same level.

Levels of decision making and information

Each decision made in a project environment should be based on adequate information of the correct sort. The type of information needed depends on the level of decision making. Decisions can be grouped at three levels: *strategic*, *tactical*, and *operational*.

Strategic decision making is essentially about deciding objectives. In the case of the Brightmouth College payroll, the decision to become administratively independent could be regarded as a strategic decision. In our example we were interested only in the payroll, but this might have been part of a wider programme which may have affected many other administrative functions.

Tactical decision making is needed to ensure that the objectives will be fulfilled. The project leader who has the responsibility for achieving objectives will have to formulate a plan of action to meet those objectives. The project leader will need to monitor progress to see whether these objectives are likely to be met and to take action where needed to ensure that the things remain on course.

Operational decisions relate to the day-to-day work of implementing the project. Deciding the content of the acceptance tests might come under this heading.

Differences in types of information

Table 1.1 gives some idea of the differences in the kind of information needed. There is a kind of continuum for most of the qualities suggested and what is needed for tactical decision making comes somewhere in the middle.

Effectiveness is concerned with doing the right thing. *Efficiency* is carrying out a task making the best possible use of the resources.

Measurement

The quantification of measures of effectiveness reduces ambiguity.

The leader of a small project will have direct contact with many aspects of the project. With larger projects, project leaders would have to depend on information being supplied to them. This information should not be vague and ideally should be quantitative. This ties in with our need for unambiguous measures of effectiveness.

Software development deals largely with intangibles and does not easily lend itself to quantitative measures, but attempts are increasingly being made to introduce measurement into the software process.

Table 1.1 *The types of information required for decision making*

Characteristic	*Operational*	*Strategic*
motivation	efficiency	effectiveness
orientation	internal	internal and external
focus	specific to a function	specific to organization
detail	detailed	summarized
response	fast	not so fast
access paths	standard	flexible
up-to-dateness	essential	desirable
accuracy	essential	approximate
certainty	essential	often predictive
objectivity	high	more subjective
information type	mainly quantitative	often qualitative

Software measurements can be divided into *performance measures* and *predictive measures*.

- **Performance measures** These measure the characteristics of a system that has been delivered. They are important when we are trying to specify unambiguously the quality requirements of a proposed system.
- **Predictive measures** The trouble with performance measures is that you need to have a system actually up and running before you can take measurements. As a project leader, what you want to be able to do is to get some idea of the likely characteristics of the final system during its development. *Predictive measures* are taken during development and indicate what the performance of the final system is likely to be.

Performance measures include mean time between failures (reliability) and time to learn an application (usability).

For example, the errors found per KLOC (that is, thousand lines of code) at different stages of the project might help to predict the correctness and reliability of the final system. Keystrokes required to carry out a particular transaction might help predict what the operator time to carry out the transaction is likely to be. Modularity, the degree to which the software is composed of self-contained manageable components, helps predict how easy changes to the final system will be.

1.13 Conclusion

This chapter has laid a foundation for the remainder of the book by defining what is meant by various terms such as ‘software project’ and ‘management’. Among some of the more important points that have been made are the following.

- Projects are by definition non-routine and therefore more uncertain than normal undertakings.

- Software projects are similar to other projects, but have some attributes that present particular difficulties, for example, the relative invisibility of many of their products.
- A key factor in project success is having clear objectives. Different stakeholders in a project, however, are likely to have different objectives. This points to the need for a recognized overall project authority.
- For objectives to be effective, there must be practical ways of testing that the objectives have been met. Hence there is a need for measurement.
- Where projects involve many different people, effective channels of information have to be established. Having objective measures of success helps unambiguous communication among the various parties to a project.

1.14 Further exercises

1. List the problems you experienced when you carried out a recent IT-related assignment. Try to put these problems into some order of magnitude. For each problem, consider whether there was some way in which the problem could have been reduced by better organization and planning by yourself.
2. Identify the main types of personnel employed in an information systems department. For each stage of a typical IS development project, list the types of personnel who are likely to be involved.
3. A public library department is considering the implementation of a computer-based system to help administer book loans at libraries. Identify the stakeholders in such a project. What might be the objectives of such a project and how might the success of the project be measured in practical terms?
4. A manager is in charge of a sub-project of a larger project. The sub-project requires the transfer of paper documents into a computer-based document retrieval system and their subsequent indexing so that they can be accessed via key-words. Optical character readers are to be used for the initial transfer but the text then needs to be clerically checked and corrected by staff. The project is currently scheduled to take twelve months using permanent staff. A small budget is available to hire temporary staff in the case of staff absences through holidays, sickness or temporary transfer to other, more urgent, jobs. Discuss the control system that will be needed to control that sub-project.
5. In the above example, concerning document transfer and indexing, identify the strategic information that the manager might want to consider during the initial planning of the sub-project.
6. Suggest objectives for the following types of staff: (a) a data preparation clerk; (b) a programmer/analyst; (c) a computer software sales person; (d) a systems analyst; (e) a software project leader. In each case suggest two measures of efficiency or effectiveness.

Chapter 2

Step Wise: an overview of project planning

OBJECTIVES

When you have completed this chapter you will be able to:

- approach project planning in an organized step-by-step manner;
- see where the techniques described in other chapters fit into an overall planning approach;
- repeat the planning process in more detail for sets of activities within a project as the time comes to execute them.

2.1 Introduction to Step Wise project planning

This chapter describes a framework of basic steps in project planning and control upon which the following chapters build. There are many different techniques that can be used in project planning and this chapter gives an overview of the points at which these techniques can be used during project planning. Chapter 4 will illustrate how different projects need different approaches, but this framework should always apply to the planning process used.

The framework described is called the Step Wise method to help to distinguish it from other methods such as PRINCE 2. PRINCE 2 is the set of project management standards that have been published by the Central Computing and Telecommunications Agency (CCTA) for use on British government IT projects. The standards are also widely used on non-government projects in the United Kingdom. Step Wise should be compatible with PRINCE 2. It should be noted, however, that Step Wise covers only the planning stages of a project and not monitoring and control.

Appendix A adds some further details about the PRINCE 2 approach. There is also some use of PRINCE 2 in the Netherlands and Australia.

In order to illustrate the Step Wise approach and to show how it might have to be adapted to deal with different circumstances, two parallel examples are used.

Let us assume that there are two former Computing and Information Systems students who have now had several years of software development experience.

Case study examples: Brightmouth College Payroll and International Office Equipment Group Maintenance Accounts

Brigette has been working for the Management Services department of a local government authority when she sees an advertisement for the position of Information Systems Development Officer at Brightmouth College. She is attracted to the idea of being her own boss, working in a relatively small organization and helping it to set up appropriate information systems from scratch. She applies for the job and gets it. One of the first tasks that confronts her is the implementation of independent payroll processing! (This scenario has already been used as the basis of some examples in Chapter 1.)

Amanda works for International Office Equipment (IOE), which manufactures and supplies various items of high-technology office equipment. An expanding area of their work is the maintenance of IT equipment. They have now started to undertake maintenance of equipment for which they were not originally the suppliers. A computer-based batch processing system deals with invoicing on a job-by-job basis. An organization might have to call IOE out several times to deal with different bits of equipment and there is a need to be able to group the invoice details for work done into 'group accounts' for which monthly statements will be produced. Amanda has been given her first project management role, the task of implementing this extension to the invoicing system.

In Table 2.1 we outline the general approach that might be taken to planning these projects. Figure 2.1 provides an outline of the main planning activities. Steps 1 and 2, 'Identify project scope and objectives' and 'Identify project infrastructure', may be tackled in parallel in some cases. Steps 5 and 6 will have to be repeated for each activity needed to complete the project.

A major principle of project planning is to plan in outline first and then in more detail as the time to carry out an activity approaches. Hence the lists of products and activities that are the result of Step 4 will be reviewed when the tasks connected with a particular phase of a project are considered in more detail. This will be followed by a more detailed iteration of Steps 5 to 8 for the phase under consideration.

2.2 Step 0: Select project

Chapter 3 discusses project evaluation in more detail.

This is called Step 0 because in a way it is outside the main project planning process. Projects are not initiated out of thin air – some activity has to take place before deciding that this project rather than another is worth undertaking. This project evaluation may be done on an individual basis or as part of strategic planning.

Table 2.1 *An outline of Step Wise planning activities*

Step	*Activities within step*
0	Select project
1	Identify project scope and objectives
	1.1 Identify objectives and measures of effectiveness in meeting them
	1.2 Establish a project authority
	1.3 Identify all stakeholders in the project and their interests
	1.4 Modify objectives in the light of stakeholder analysis
	1.5 Establish methods of communications with all parties
2	Identify project infrastructure
	2.1 Establish relationship between project and strategic planning
	2.2 Identify installation standards and procedures
	2.3 Identify project team organization
3	Analyse project characteristics
	3.1 Distinguish the project as either objective- or product-driven
	3.2 Analyse other project characteristics
	3.3 Identify high level project risks
	3.4 Take into account user requirements concerning implementation
	3.5 Select general lifecycle approach
	3.6 Review overall resource estimates
4	Identify project products and activities
	4.1 Identify and describe project products (or deliverables)
	4.2 Document generic product flows
	4.3 Recognize product instances
	4.4 Produce ideal activity network
	4.5 Modify ideal to take into account need for stages and checkpoints
5	Estimate effort for each activity
	5.1 Carry out bottom-up estimates
	5.2 Revise plan to create controllable activities
6	Identify activity risks
	6.1 Identify and quantify activity-based risks
	6.2 Plan risk reduction and contingency measures where appropriate
	6.3 Adjust plans and estimates to take account of risks
7	Allocate resources
	7.1 Identify and allocate resources
	7.2 Revise plans and estimates to account for resource constraints
8	Review/publicize plan
	8.1 Review quality aspects of project plan
	8.2 Document plans and obtain agreement
9	Execute plan
10	Lower levels of planning

Figure 2.1 will be revisited in subsequent chapters where we will highlight and describe the individual steps in the Step Wise framework.

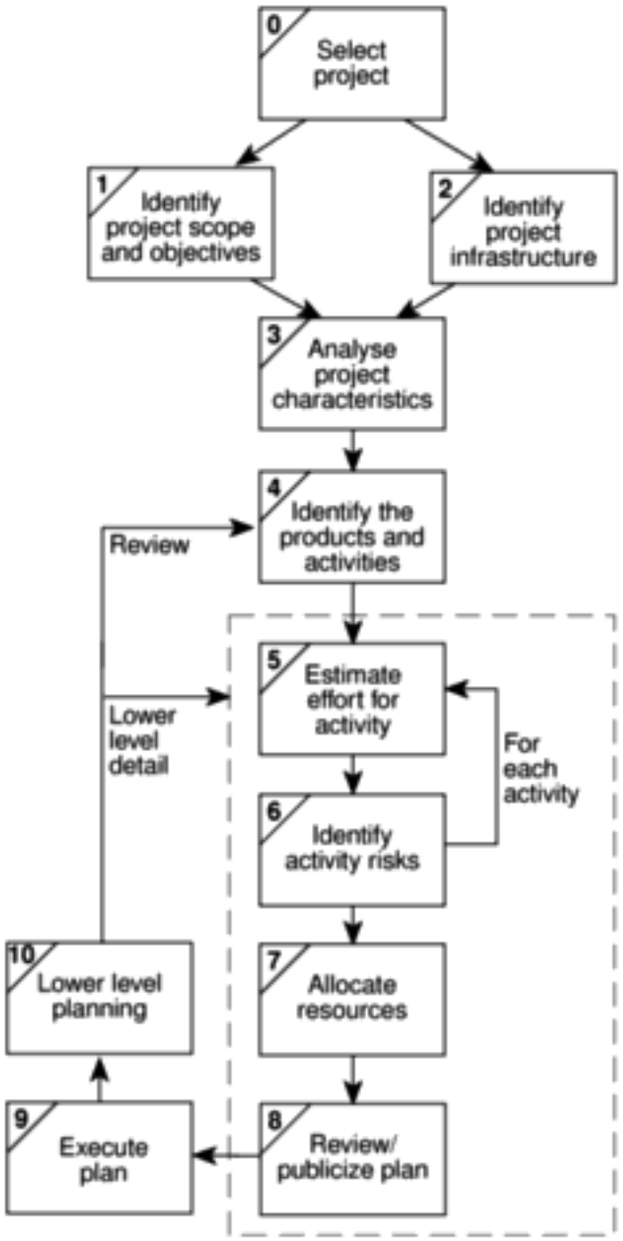


Figure 2.1 *An overview of Step Wise.*

2.3 Step 1: Identify project scope and objectives

The activities in this step ensure that all the parties to the project agree on the objectives and are committed to the success of the project. A danger to be avoided is overlooking people who are affected by the project.

Step 1.1: Identify objectives and practical measures of the effectiveness in meeting those objectives

We discussed earlier the need for agreed objectives for a project and ways of measuring the success in achieving those objectives.

Case Study Example: Project objectives

The project objectives for the Brightmouth College payroll project have already been discussed in Exercise 1.7.

Amanda at IOE has the objectives clearly laid down for her in the recommendations of a feasibility study report, which have been accepted by IOE management. The main objectives are to allow a detailed monthly statement to be sent to group account clients and to be able to reallocate the cash received to individual jobs when the client has paid on the monthly statement. There are also other objectives that refer to expected timescales and the resources that may be used.

Step 1.2: Establish a project authority

A single overall project authority needs to be established so that there is unity of purpose among all those concerned.

Case Study Examples: Project authorities

Amanda finds that her manager and the main user management have already set up a Project Board that will have overall direction of the project. She is a little concerned because the equipment maintenance staff are organized with different sections dealing with different types of equipment. This means that a customer might have work done by several different sections. Not all the sections are represented on the Project Board and Amanda is aware that there are some differences of opinion among the different sections. It is left to the user representatives on the board to resolve those differences and to present an agreed policy to the systems developers.

Throughout the text we use capitalized initial letters to indicate a term that has a precise meaning in the PRINCE 2 standards, e.g. Project Board.

Brigette finds that effectively she has two different clients for the payroll system: the finance and personnel departments. To help resolve conflicts, it is agreed that the managers of both departments should attend a monthly meeting with the Vice-Principal, which Brigette has arranged in order to steer the project.

Step 1.3: Identify all stakeholders in the project and their interests

Recall that this was the basis of a discussion in Chapter 1. Essentially all the parties who have an interest in the project need to be identified. In Exercise 1.8 you produced a list of the stakeholders in the Brightmouth College Payroll project.

Exercise 2.1

What important stakeholders outside the IOE organization should be considered in the case of the IOE Maintenance Group Accounts System?

Step 1.4: Modify objectives in the light of stakeholder analysis

In order to gain the full cooperation of all concerned, it might be necessary to modify the project objectives. This can mean adding new features to the system giving a benefit to some stakeholder group as a means of assuring their commitment to the project. This is potentially dangerous, since the system size might be increased and the original objectives obscured. Because of these dangers, this process must be done consciously and in a controlled manner.

Case Study Examples: Modified project objectives

The IOE maintenance staff are to be given the extra task of entering data about completed jobs. They do not benefit from this additional work. To give some benefit, the system is to be extended to reorder spare parts automatically when required.

At Brightmouth College, the personnel department has a lot of work preparing payroll details for finance. It will be tactful to agree to produce some management information reports for personnel from the payroll details held on the computer.

Step 1.5: Establish methods of communication with all parties

For internal staff, this should be fairly straightforward, but a project leader implementing a payroll system would need to find a contact point with BACS (Bankers Automated Clearing Scheme) for instance.

2.4 Step 2: Identify project infrastructure

Projects are rarely initiated in a vacuum. There is usually some kind of existing infrastructure into which the project can fit. The project leader who does not already know about this structure needs to find out its precise nature.

Step 2.1: Identify relationship between the project and strategic planning

Some of the issues of strategic planning are addressed in Chapter 3.

As well as identifying projects to be carried out, an organization needs to decide the order in which these projects are to be carried out. It also needs to establish the framework within which the proposed new systems are to fit. Hardware and software standards, for example, are needed so that various systems can communicate with each other. These strategic decisions must be documented in a strategic business plan or in an information technology plan that is developed from the business plan.

Case Study Examples: Role of existing strategic plans

Amanda finds at IOE that there is a well-defined rolling strategic plan that has identified her group accounts subsystem as an important required development. Because it is an extension of an existing system, the hardware and software platforms upon which the application are to run are dictated.

Brigette at Brightmouth College finds that there is an overall College strategic plan that describes new courses to be developed, and so on, and mentions in passing the need for 'appropriate administrative procedures' to be in place. In a short section in a consultant's report from an accountancy firm concerning the implications of financial autonomy, there is a recommendation that independent payroll processing be undertaken. Although the college has quite a lot of IT equipment for teaching purposes, there is no machine set aside for payroll processing and the intention is that the hardware to run the payroll will be acquired at the same time as the software.

Step 2.2: Identify installation standards and procedures

Any organization that develops software should define its development procedures. As a minimum, the normal stages in the software life cycle to be carried out should be documented along with the products created at each stage.

Change control and *configuration management* standards should be in place to ensure that changes to requirements are implemented in a safe and orderly way.

See Chapter 9 on Monitoring and Control.

The procedural standards may lay down the quality checks that need to be done at each point of the project life cycle or these may be documented in a separate *quality standards and procedures* manual.

The organization, as part of its monitoring and control policy must have in place a *measurement programme* that dictates that certain statistics have to be collected at various stages of a project.

Finally the project manager should be aware of any *project planning and control standards*. These will relate to the way that the project is controlled: for example, the way that the hours spent by team members on individual tasks are recorded on time-sheets

---

Case Study Examples: Identifying standards

Amanda at IOE finds that there is a very weighty manual of development standards, which, among other things, specifies that SSADM will be the analysis and design method used. She finds that a separate document has been prepared, laying down quality procedures. This specifies when the reviews of work will be carried out and describes detailed procedures about how the reviews are to be done. Amanda also finds a set of project management guidelines modelled closely on PRINCE 2.

Brigette finds no documents of the nature that Amanda found at IOE except for some handouts for students that have been produced by different lecturers at different times and that seem to contradict each other.

As a stop-gap measure, Brigette writes a brief document, which states what the main stages of a 'project' (perhaps 'job for the user' would be a better term in this context) should be. This happens to be very similar to the list given in Chapter 1. She stresses that:

- no job of work to change a system or implement a new one is to be done without there being a detailed specification first;
- the users must agree to, or 'sign off', each specification in writing before the work is carried out.

She draws up a simple procedure for recording all changes to user requirements.

Brigette, of course, has no organizational quality procedures, but she dictates that each person in the group (including herself) has to get someone else to check through his or her work at the end of a major task and that, before any new or amended software is handed over to the users, someone other than the original producer should test it. She sets up a simple system to record errors found in system testing and their resolution. She also creates a log file of reported user problems with operational systems.

Brigette does not worry about time sheets but arranges an informal meeting with her colleagues each Monday morning to discuss how things are going and also arranges to see the Vice-Principal, who is her official boss, and the heads of the finance and personnel sections each month to review progress in general terms.

Step 2.3: Identify project team organization

Some of these issues will be discussed in Chapter 11 – Managing people and organizing teams.

Project leaders, especially in the case of large projects, will often have some control over the organizational structure of the project team. More often, though, the organizational structure will be dictated to them. For example, there might have been a high level managerial decision that code developers and systems analysts will be in different groups, or that the development of PC applications will not be done within the same group as that responsible for 'legacy' main-frame applications.

If the project leader does have some control over the project team organization then this would best be considered at a later stage (see Step 7: Allocate resources).

Case Study Examples: Project organization

At IOE, there are groups of systems analysts set up as teams that deal with individual user departments. Hence the users always know whom they should contact within the information systems department if they have a problem. Code developers, however, work in a 'pool' and are allocated to specific projects on an *ad hoc* basis.

At Brightmouth College, Brigette has seconded to her a software developer who has been acting as a technician supporting the computing courses in the college. She is also allowed to recruit a trainee analyst/programmer. She is not unduly worried about the organizational structure that is needed.

2.5 Step 3: Analyse project characteristics

The general purpose of this part of the planning operation is to ensure that the appropriate methods are used for the project.

Chapter 4 elaborates on the process of analysing project characteristics.

Step 3.1: Distinguish the project as either objective- or product-driven

This has already been discussed in the first chapter. A general point to note is that as system development advances, it tends to become more product-driven, although the underlying objectives always remain and must be respected.

Step 3.2: Analyse other project characteristics (including quality-based ones)

For example, is this an information system that is being developed or a process control system, or does it have elements of both? Is it a safety-critical system, that is, where human life could be threatened by a malfunction?

Step 3.3: Identify high level project risks

Consideration must be given to the risks that threaten the successful outcome of the project. Generally speaking most risks can be attributed to the operational or development environment, the technical nature of the project or the type of product being created.

Case Study Example: High level risks

At IOE, Amanda identifies the danger of there being resistance to the new system by maintenance engineers, especially as a new centralized group accounts office is to be set up. Amanda decides therefore that additional efforts are needed to consult all sections involved and that the new procedures should be introduced in small increments to accustom staff to them gradually.

Brigette at Brightmouth College considers the application area to be very well-defined. There is a risk, however, that there might be no application on the market that caters for the way that things are done at the moment. Brigette, therefore, decides that an early task in the project is to obtain information about the features of the main payroll applications that are available.

Step 3.4: Take into account user requirements concerning implementation

The clients will usually have their own procedural requirements. For example, work for government departments usually requires the use of SSADM.

Step 3.5: Select general lifecycle approach in the light of the above

The project life cycle to be used for the project will be influenced by the issues raised above. For example, a prototyping approach might be used where the user requirements are not clear.

Chapter 4 discusses life cycles in more detail.

Step 3.6: Review overall resource estimates

Chapter 5 goes into more detail on this topic. Function points are an attempt to measure system size without using the number of lines of code.

Once the major risks have been identified and the broad project approach has been decided upon, this would be a good point at which to re-estimate the effort and other resources required to implement the project. Where enough information is available, an estimate based on function points might be appropriate.

2.6 Step 4: Identify project products and activities

The more detailed planning of the individual activities that will be needed now takes place. The longer term planning is broad and in outline, while the more immediate tasks are planned in some detail.

Step 4.1: Identify and describe project products (or deliverables)

Chapter 7 goes into this in more detail.

In general there can be no project products that do not have activities that create them. Wherever possible, we ought also to ensure the reverse: that there are no activities that do not produce a tangible product. Making sure we have identified all the things the project is to create helps us to ensure that all the activities we need to carry out are accounted for.

These products will include a large number of *technical* products including training material and operating instructions, but also products to do with the *management* and the *quality* of the project. Planning documents would, for example, be management products.

PRINCE 2 suggests that the PBS be presented as a hierarchy diagram. In practice it may be more convenient to produce a structured list.

The products will form a hierarchy. The main products will have sets of component products, which in turn might have sub-component products and so on. These relationships can be documented in a Product Breakdown Structure (PBS).

This part of the planning process draws heavily on the standards laid down in PRINCE 2. These specify that products at the bottom of the PBS should be documented by *Product Descriptions*, which contain:

- the name/identity of the product;
- the purpose of the product;
- the derivation of the product (that is, the other products from which it is derived);
- the composition of the product;
- the form of the product;
- the relevant standards;
- the quality criteria that should apply to it.

Case Study Example: PBS

At IOE, Amanda finds that there is a standard PBS that she can use as a check-list for her own project.

Brigette at Brightmouth College has no installation standard PBS, although she can, of course, refer to various books for standard check-lists. She decides that one part of the PBS should contain the products needed to help select the appropriate hardware and software for the payroll application (Figure 2.2).

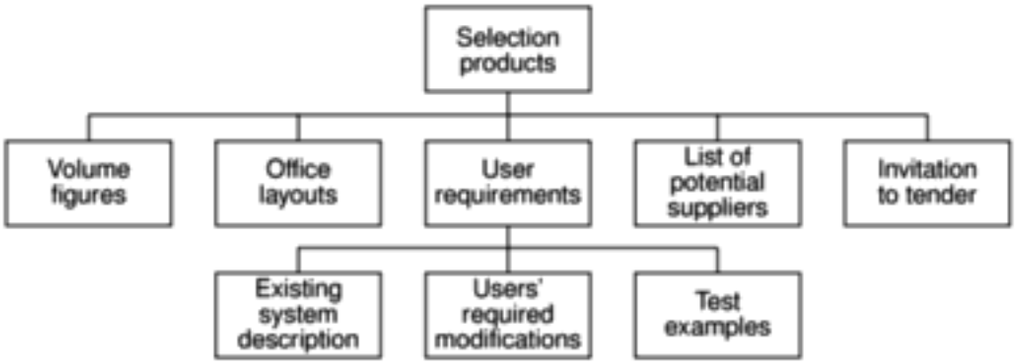


Figure 2.2 *A fragment of the PBS for the Brightmouth College Payroll Project.*

Step 4.2: Document generic product flows

Some of the products will need some other product to exist first before they can be created. For example, a program design must be created before the program can be written and the program specification must exist before the design can be commenced. These relationships can be portrayed in a Product Flow Diagram (PFD). Figure 2.3 gives an example.

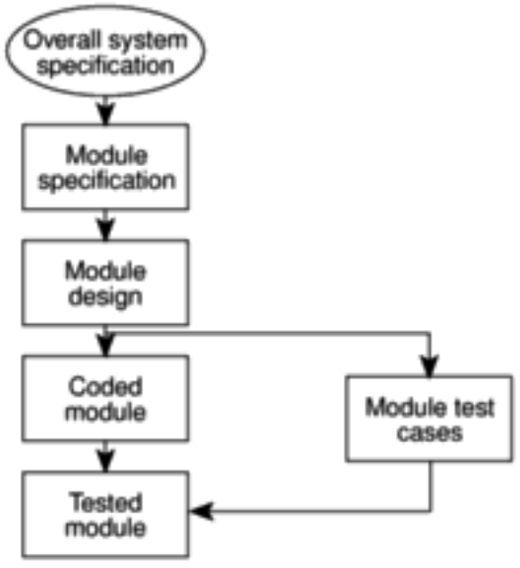


Figure 2.3 *A fragment of a PFD.*

Case Study Example: IOE has standard PFD

At IOE, Amanda has a standard installation PFD. Many of the products that will make up Amanda's application will be of the same type: hence the same generic PFD will apply to each instance. It is pointless in these circumstances to draw up a separate PFD for each instance of the product.

Exercise 2.2

Draw up a possible Product Flow Diagram (PFD) based on the Product Breakdown Structure (PBS) shown in Figure 2.2. This represents the products that will be produced when gathering information to be presented to potential suppliers of the hardware. The volume figures are for such things as the number of employees for whom records will have to be maintained.

This may be delayed to later in the project when more information is known.

Step 4.3: Recognize product instances

Where the same generic PFD fragment relates to more than one instance of a particular type of product, an attempt should be made to identify each of those instances.

Case Study Example: identifying product instances

Amanda decides that there are likely to be four major software modules needed in her application for which the PFD fragment in Figure 2.3 would be appropriate

The products that Brigette can identify at the present all have a single instance.

Step 4.4: Produce ideal activity network

In order to generate one product from another there must be one or more activities that carry out the transformation. By identifying these activities we can create an activity network, which shows the tasks that have to be carried out and the order in which they have to be executed.

Case Study Example: Activity network for IOE Maintenance Accounts

Part of the initial activity network for the IOE Maintenance Group Accounts project might look like Figure 2.4.

Exercise 2.3

Draw up an Activity Network for the Product Flow Diagram that you created in Exercise 2.2 (or the PFD given in the solution if you prefer!).

The activity networks are 'ideal' in the sense that no account has been taken of resource constraints. For example in Figure 2.4, it is assumed that resources are available for all four software modules to be developed in parallel.

Step 4.5: Modify the ideal to take into account need for stages and checkpoints

The approach to sequencing activities described above encourages the formulation of a plan that will minimize the overall duration, or 'elapsed time', for the project. It assumes that an activity will start as soon as the preceding ones upon which it depends have been completed.

There might, however, be a need to modify this by dividing the project into stages and introducing checkpoint activities. These are activities that draw together the products of preceding activities to check that they are compatible. These checkpoints are sometimes referred to as milestone events. A checkpoint could potentially delay work on some elements of the project – there has to be a trade-off between efficiency and quality.

Exercise 2.4

Amanda decides that after the four modules have been specified, the four specifications need to be carefully checked to see that they are consistent and compatible. Redraw the activity network in Figure 2.4 to reflect this.

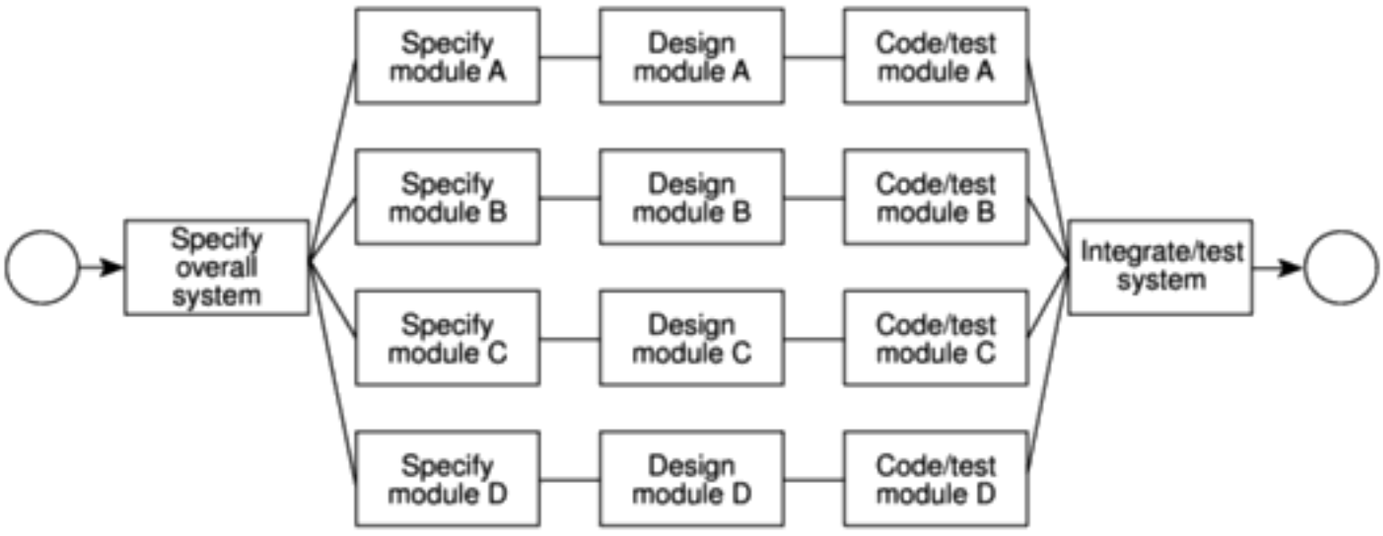


Figure 2.4 *An activity network fragment for the IOE Maintenance Group Accounts project.*

2.7 Step 5: Estimate effort for each activity

Step 5.1: Carry out bottom-up estimates

Some top-down estimates of effort, cost and duration will already have been done (see Step 3.6).

Chapter 5 on Software Estimation deals with this topic in more detail.

At this point, estimates of the staff effort and other resources required, and the probable elapsed time needed for each activity will need to be produced. The method of arriving at each of these estimates will vary depending on the type of activity.

The individual activity estimates of effort should be summed to get an overall bottom-up estimate, which can be reconciled with the previous top-down estimate.

The activities on the activity network can be annotated with their elapsed times so that the overall duration of the project can be calculated.

Step 5.2: Revise plan to create controllable activities

The estimates for individual activities might reveal that some are going to take quite a long time. Long activities often make a project difficult to control. If an activity involving system testing is to take 12 weeks, it might be difficult after six weeks to judge accurately whether 50% of the work is completed. It would be better to break this down into a series of smaller sub-tasks.

Case Study Example: IOE Maintenance Group Accounts – breaking activities down into manageable tasks

At IOE, Amanda has to estimate the lines of code for each of the software modules. She looks at programs that have been coded for similar types of application at IOE in the past to get some idea of the size of the new modules. She then refers to some conversion tables that the information systems development department at IOE have produced; these tables convert the lines of code into estimates of effort. Other tables allow her to allocate the estimated effort to the various stages of the project.

Although Brigette is aware that some additional programs might have to be written to deal with local requirements, the main software is to be obtained 'off-the-shelf' and so estimating based on lines of code would clearly be inappropriate. Instead, she looks at each individual task and allocates a time. She realizes that in many cases these represent 'targets' as she is uncertain at the moment how long these tasks will really take (see Step 6 below).

2.8 Step 6: Identify activity risks

Step 6.1: Identify and quantify activity-based risks

Chapter 7 on Risk touches on this topic in more detail.

Risks inherent in the overall nature of the project have already been considered in Step 3. We now want to look at each activity in turn and assess the risks to its successful outcome. The seriousness of each risk and likelihood of it occurring

have to be gauged. At individual task level some risks are unavoidable, and the general effect if a problem materializes is to make the task longer or more costly. A range of estimates can be produced to take into account the possible occurrence of the risks.

Step 6.2: Plan risk reduction and contingency measures where appropriate

It is possible to avoid or at least reduce some of the identified risks. Contingency plans specify action that is to be taken if a risk materializes. For example, a contingency plan could be to use contract staff if a member of the project team is unavailable at a key time because of illness.

Step 6.3: Adjust overall plans and estimates to take account of risks

We can change our plans, perhaps by adding new activities which reduce risks. For example, a new programming language could mean that we schedule training courses and time for the programmers to practise their new programming skills on some non-essential work.

Case Study Example: Identifying risks

As well as the four new software modules that will have to be written, Amanda has identified several existing modules that will need to be amended. The ease with which the modules can be amended will depend upon the way in which they were originally written. There is therefore a risk that they will take longer than expected to modify. Amanda takes no risk reduction measures as such but notes a pessimistic elapsed time for the amendment activity.

Brigette identifies as a risk the possible absence of key staff when investigating the user requirements as this activity will take place over the holiday period. To reduce this risk, she adds a new activity, 'arrange user interviews', at the beginning of the project. This will give her advance notice of any likely problems of this nature.

2.9 Step 7: Allocate resources

Step 7.1: Identify and allocate resources

Chapter 8 on Resource allocation covers this topic in more detail.

The type of staff needed for each activity is recorded. The staff available for the project are identified and are provisionally allocated to tasks.

Step 7.2: Revise plans and estimates to take into account resource constraints

Some staff might be needed for more than one task at the same time and, in this case, an order of priority is established. The decisions made here can have an effect on the overall duration of the project when some tasks are delayed while waiting for staff to become free.

Ensuring someone is available to start work on an activity as soon as the preceding activities have been completed might mean that they are idle while waiting for the job to start and are therefore used inefficiently.

Case Study Example: Taking resource constraints into account

Amanda has now identified four major software modules plus two existing software modules that will need extensive amendment. At IOE, the specification of modules is carried out by the lead systems analyst for the project (who in this case is Amanda) assisted by junior analyst/designers. Four analyst/programmers are available to carry out the design, coding and unit testing of the individual modules. After careful consideration and discussion with her manager, Amanda decides to use only three analyst/programmers to minimize the risk of staff waiting between tasks. It is accepted that this decision, while reducing the cost of the project, will delay its end.

Brigette finds that she herself will have to carry out many important activities. She can reduce the workload on herself by delegating some work to her two colleagues, but she realizes that she will have to devote more time to specifying exactly what they will have to do and to checking their work. She adjusts her plan accordingly.

2.10 Step 8: Review/publicize plan

Step 8.1: Review quality aspects of the project plan

A danger when controlling any project is that an activity can reveal that an earlier activity was not properly completed and needs to be reworked. This, at a stroke, can transform a project that appears to be progressing satisfactorily into one that is badly out of control. It is important to know that when a task is reported as completed, it really is – hence the importance of quality reviews. Each task should have 'exit requirements'. These are quality checks that have to be passed before the activity can be 'signed off' as completed.

Case Study Example: IOE existing quality standards

Amanda finds that at IOE, the Quality Standards and Procedures Manual lays down quality criteria for each type of task. For example, all module design documentation has to be reviewed by a group of colleagues before the coding of that module can commence.

Exercise 2.5

Brigette has no installation standards to help her apart from the minimal ones she has written herself. What quality checks might Brigette introduce to ensure that she has understood the users' requirements properly?

Step 8.2: Document plans and obtain agreement

It is important that the plans be carefully documented and that all the parties to the project understand and agree to the commitments required of them in the plan.

2.11 Steps 9 and 10: Execute plan and Lower levels of planning

Once the project is under way, plans will need to be drawn up in greater detail for each activity as it becomes due. Detailed planning of the later stages will have to be delayed because more information will be available nearer the start of the stage. Of course, it is necessary to make provisional plans for the more distant tasks, because thinking about what has to be done can help unearth potential problems, but sight should not be lost of the fact that these plans are provisional.

Case Study Examples: Lower level planning

While work is going on with the specification of the individual modules, Amanda has some time to start planning the integration tests in some detail. She finds that, in fact, integration testing of two of the six new or amended modules will be independent of the others. Testing of these two can start when they are ready without waiting for the remainder.

When Brigette comes to consider the activity 'draft invitation to tender', she has to familiarize herself with the detailed institutional rules and procedures that govern this process. She finds that, to draft this document, she will need to obtain some additional information from the users.

2.12 Conclusion

This chapter has presented a framework into which the techniques described in the other parts of the book should slot. Any planning approach should have the following elements:

- the establishment of project objectives;
- the analysis of the characteristics of the project;
- the establishment of an infrastructure consisting of an appropriate organization and set of standards, methods and tools;
- the identification of the products of the project and the activities needed to generate those products;
- the allocation of resources to activities;
- the establishment of quality controls.

Project management is an iterative process. As the time approaches for particular activities to be carried out they should be replanned in more detail.

2.13 Further Exercises

1. List the products created by the Step Wise planning process.

2. What products must exist before the activity 'test program' can take place? What products does this activity create?

3. An employee of a training organization has the task of creating case study exercises and solutions for a training course that teaches a new systems analysis and design method. The person's work plan has a three-week task 'learn new method'. A colleague suggests that this is unsatisfactory as a task because there are no concrete deliverables or products from the activity. What can be done about this?

4. In order to carry out usability tests for a new word processing application, the software has to be written and debugged. User instructions have to be available describing how the application is to be used. These have to be scrutinized in order to plan and design the tests. Subjects who will use the application in the tests will need to be selected. As part of this selection process, they will have to complete a questionnaire giving details of their past experience of, and training in, typing and using word processing applications. The subjects will carry out the required tasks using the word processing application. The tasks will be timed and any problems the subjects encounter with the application will be noted. After the test, the subjects will complete another questionnaire about what they felt about the application. All the data from the tests will be analysed and a report containing recommendations for changes to the application will be drawn up. Draw up a Product Breakdown Structure, a Product Flow Diagram and a preliminary activity network for the above.

5. Identify the actions that could prevent each of the following risks from materializing or could reduce the impact if it did occur:

   (a) a key member of the programming team leaving;

   (b) introducing a new version of the operating system that has errors in it;

   (c) a disk containing copies of the most up-to-date version of the software under development being corrupted;

   (d) system testing unearthing more errors than were expected and taking longer than planned;

   (e) the government changes the taxation rules, altering the way that Value Added Tax (VAT) is to be calculated in an order processing system under development.

6. Read Appendix A on PRINCE 2. To what extent could the PRINCE 2 project management standards be usefully applied to the Brightmouth College payroll?

Chapter 3

Project evaluation

OBJECTIVES

When you have completed this chapter you will be able to:

- carry out an evaluation and selection of projects against strategic, technical and economic criteria;
- use a variety of cost-benefit evaluation techniques for choosing among competing project proposals;
- evaluate the risk involved in a project and select appropriate strategies for minimizing potential costs.

3.1 Introduction

Deciding whether or not to go ahead with a project is really a case of comparing a proposed project with the alternatives and deciding whether to proceed with it. That evaluation will be based on strategic, technical and economic criteria and will normally be undertaken as part of strategic planning or a feasibility study for any information system development. The risks involved also need to be evaluated.

'Do nothing' is an option which should always be considered.

In this chapter we shall be using the term *project* in a broader sense than elsewhere in the book. Our decision as to whether or not to proceed with a project needs to be based upon whether or not it is desirable to carry out the development *and operation* of a software system. The term project may therefore be used, in this context, to describe the whole life cycle of a system from conception through to final decommissioning.

The BS 6079 guidelines (see Appendix B) use the term project in this way.

Project evaluation is normally carried out in Step 0 of Step Wise (Figure 3.1). The subsequent steps of Step Wise are then concerned with managing the development project that stems from this project selection.

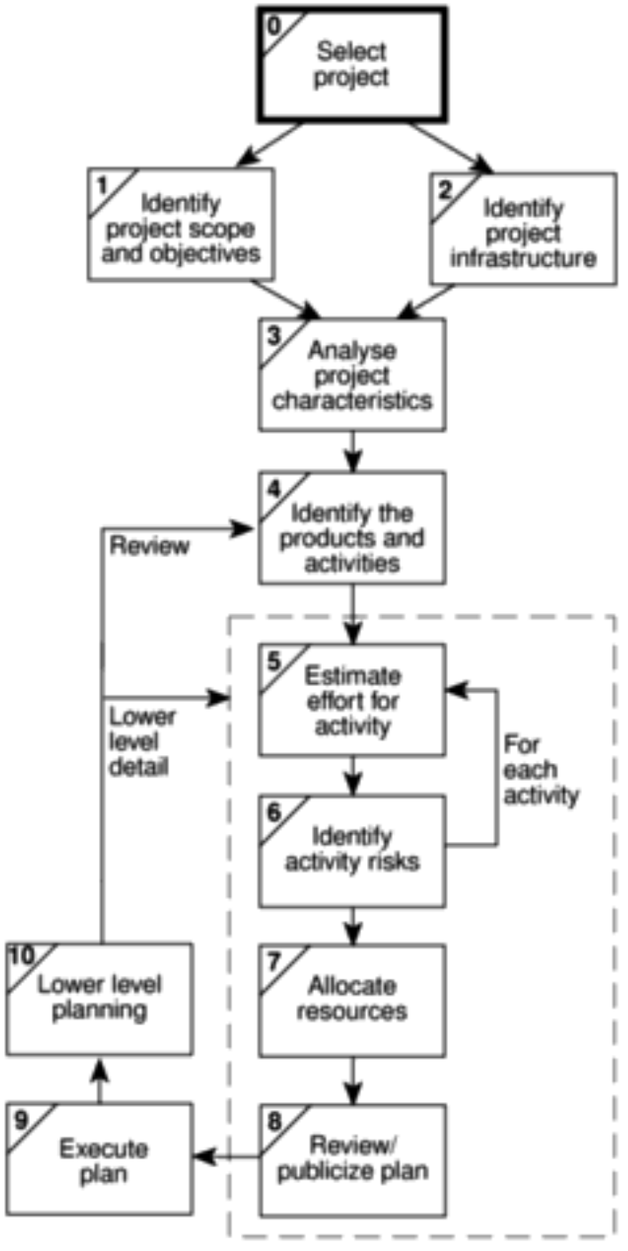


Figure 3.1 *Project evaluation is carried out in Step 0.*

D. C. Ferns defined a programme as 'a group of projects that are managed in a co-ordinated way to gain benefits that would not be possible were the projects to be managed independently' in a seminal article in the *International Journal of Project Management*, August 1991.

3.2 Strategic assessment

Programme management

It is being increasingly recognized that individual projects need to be seen as components of a *programme* and should be evaluated and managed as such. A programme, in this context, is a collection of projects that all contribute to the same overall organizational goals. Effective programme management requires that there is a well defined *programme goal* and that all the organization's projects are selected and tuned to contribute to this goal. A project must be evaluated according to how it contributes to this programme goal and its viability, timing, resourcing and final worth can be affected by the programme as a whole. It is to be expected

that the value of any project is increased by the fact that it is part of a programme – the whole, as they say, being greater than the sum of the parts.

In order to carry out a successful strategic assessment of a potential project there should therefore be a strategic plan clearly defining the organization's objectives. This provides the context for defining the programme and programme goals and, hence, the context for assessing the individual project. It is likely, particularly in a large organization, that there will be an organizational structure for programme management and it will be, for example, the *programme director* and *programme executive*, rather than, say, a project manager, who will be responsible for the strategic assessment of a proposed project.

Even where there is no explicitly defined programme, any proposed project must be evaluated within the context of the organization's overall business objectives. Moreover, any potential software system will form part of the user organization's overall information system and must be evaluated within the context of the existing information system and the organization's information strategy. Table 3.1 illustrates typical issues that must be addressed as part of the strategic assessment of a project.

Table 3.1 *Typical issues and questions to be considered during strategic assessment*

Issue	*Typical questions*
Objectives	How will the proposed system contribute to the organization's stated objectives? How, for example, might it contribute to an increase in market share?
IS plan	How does the proposed system fit into the IS plan? Which existing system(s) will it replace/interface with? How will it interact with systems proposed for later development?
Organization structure	What effect will the new system have on the existing departmental and organization structure? Will, for example, a new sales order processing system overlap existing sales and stock control functions?
MIS	What information will the system provide and at what levels in the organization? In what ways will it complement or enhance existing management information systems?
Personnel	In what way will the proposed system affect manning levels and the existing employee skill base? What are the implications for the organization's overall policy on staff development?
Image	What, if any, will be the effect on customers' attitudes towards the organization? Will the adoption of, say, automated systems conflict with the objectives of providing a friendly service?

Where a well-defined information systems strategy does not exist, system development and the assessment of project proposals will be based on a more piecemeal approach – each project being individually assessed early in its life cycle. In such cases it is likely that cost–benefit analysis will have more importance and some of the questions of Table 3.1 will be more difficult to answer.

Portfolio management

Third party developers must also carry out strategic and operational assessment of project proposals.

Where an organization such as a software house is developing a software system they could be asked to carry out a strategic and operational assessment on behalf of the customer. Whether or not this should be the case, they will require an assessment of any proposed project themselves. They will need to ensure that carrying out the development of a system is consistent with their own strategic plan – it is unlikely, for example, that a software house specializing in financial and accounting systems would wish to undertake development of a factory control system unless their strategic plan placed an emphasis on diversification.

The proposed project will form part of a *portfolio* of ongoing and planned projects and the selection of projects must take account of the possible effects on other projects in the portfolio (competition for resources, for example) and the overall portfolio profile (for example, specialization versus diversification).

3.3 Technical assessment

Technical assessment of a proposed system consists of evaluating the required functionality against the hardware and software available. Where an organization has a strategic information systems plan, this is likely to place limitations on the nature of solutions that might be considered. The constraints will, of course, influence the cost of the solution and this must be taken into account in the cost–benefit analysis.

3.4 Cost–benefit analysis

The most common way of carrying out an economic assessment of a proposed information system, or other development, is by comparing the expected costs of development and operation of the system with the benefits of having it in place.

Any project requiring an investment must, as a minimum, provide a greater benefit than putting that investment in, say, a bank.

Assessment is based upon the question of whether the estimated costs are exceeded by the estimated income and other benefits. Additionally, it is usually necessary to ask whether or not the project under consideration is the best of a number of options. There might be more candidate projects than can be undertaken at any one time and, in any case, projects will need to be prioritized so that any scarce resources may be allocated effectively.

The standard way of evaluating the economic benefits of any project is to carry out a cost–benefit analysis, which consists of two steps.

- **Identifying and estimating all of the costs and benefits of carrying out the project** This includes development costs of the system, the operating costs and the benefits that are expected to accrue from the operation of the system. Where the proposed system is replacing an existing one, these estimates should reflect the costs and benefits due to the new system. A sales order processing system, for example, could not claim to benefit an organization by the total value of sales – only by the increase due to the use of the new system.
- **Expressing these costs and benefits in common units** We must evaluate the net benefit, which is the difference between the total benefit and the total cost. To do this, we must express each cost and each benefit in monetary terms.

Most costs are relatively easy to identify and quantify in approximate monetary terms. It is helpful to categorize costs according to where they originate in the life of the project.

Many costs are easy to identify and measure in monetary terms.

- **Development costs** – include the salaries and other employment costs of the staff involved in the development project and all associated costs.
- **Setup costs** – include the costs of putting the system into place. These consist mainly of the costs of any new hardware and ancillary equipment but will also include costs of file conversion, recruitment and staff training.
- **Operational costs** – consist of the costs of operating the system once it has been installed.

Benefits, on the other hand, are often quite difficult to quantify in monetary terms even once they have been identified. Benefits may be categorized as follows.

Indirect benefits, which are difficult to estimate, are sometimes known as intangible benefits.

- **Direct benefits** – these accrue directly from the operation of the proposed system. These could, for example, include the reduction in salary bills through the introduction of a new, computerized system.
- **Assessable indirect benefits** – these are generally secondary benefits, such as increased accuracy through the introduction of a more user-friendly screen design where we might be able to estimate the reduction in errors, and hence costs, of the proposed system.
- **Intangible benefits** – these are generally longer term or benefits that are considered very difficult to quantify. Enhanced job interest can lead to reduced staff turnover and, hence, lower recruitment costs.

Exercise 3.1

Brightmouth College are considering the replacement of the existing payroll service, operated by a third party, with a tailored, off-the-shelf computer-based system. List some of the costs and benefits they might consider under each of the six headings given above. For each cost or benefit, explain how, in principle, it might be measured in monetary terms.

If a proposal shows an excess of benefits over costs then it is a candidate for further consideration.

Any project that shows an excess of benefits over costs is clearly worth considering for implementation. However, as we shall see later, it is not a sufficient justification for going ahead: we might not be able to afford the costs; there might be even better projects we could allocate our resources to instead; the project might be too risky.

3.5 Cash flow forecasting

As important as estimating the overall costs and benefits of a project is the forecasting of the cash flows that will take place and their timing. A cash flow forecast will indicate when expenditure and income will take place (Figure 3.2).

Typically products generate a negative cash flow during their development followed by a positive cash flow over their operating life. There might be decommissioning costs at the end of a product's life

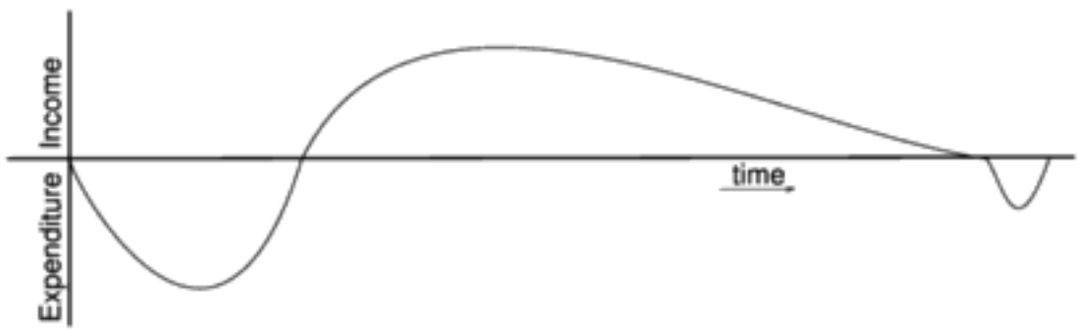


Figure 3.2 *Typical product life cycle cash flow.*

We need to spend money, such as staff wages, during the development stages of a project. Such expenditure cannot be deferred until income is received (either from using the software if it is being developed for in-house use or from selling it). It is important that we know that we can fund the development expenditure either from the company's own resources or by borrowing from the bank. In any event, it is vital to have some forecast of when expenditure such as the payment of salaries and bank interest will take place and when any income is to be expected, such as payment on completion or, possibly, stage payments.

The difficulty and importance of cash flow forecasting is evidenced by the number of companies that suffer bankruptcy because, although they are developing profitable products or services, they cannot sustain an unplanned negative cash flow.

Accurate cash flow forecasting is not easy, as it generally needs to be done early in the project's life cycle (at least before any significant expenditure is committed) and many items to be estimated (particularly the benefits of using software or decommissioning costs) might be some years in the future.

When estimating future cash flows, it is usual to ignore the effects of inflation. Trying to forecast the effects of inflation increases the uncertainty of the forecasts. Moreover, if expenditure is increased due to inflation it is likely that income will increase proportionately. However, measures to deal with increases in costs where work is being done for an external customer must be in place – for example index linked prices where work involves use of raw materials – see Chapter 10 on contract management.

Table 3.2 illustrates cash flow forecasts for four projects. In each case it is assumed that the cash flows take place at the end of each year. For short-term projects or where candidate projects demonstrate significant seasonal cash flow

patterns it can be advisable to produce quarterly, or even monthly, cash flow forecasts.

3.6 Cost–benefit evaluation techniques

We would consider proceeding with a project only where the benefits outweigh the costs. However, in order to choose among projects, we need to take into account the timing of the costs and benefits as well as the benefits relative to the size of the investment.

Exercise 3.2

Consider the project cash flow estimates for four projects at IOE shown in Table 3.2. Negative values represent expenditure and positive values income.

Rank the four projects in order of financial desirability and make a note of your reasons for ranking them in that way before reading further.

In the following sections we will take a brief look at some common methods for comparing projects on the basis of their cash flow forecasts.

Net profit

The net profit of a project is the difference between the total costs and the total income over the life of the project. Project 2 in Table 3.2 shows the greatest net profit but this is at the expense of a large investment. Indeed, if we had £1m to invest, we might undertake all of the other three projects and obtain an even greater net profit. Note also, that all projects contain an element of risk and we might not be prepared to risk £1m. We shall look at the effects of risk and investment later in this chapter.

Table 3.2 *Four project cash flow projections – figures are end of year totals (£)*

Year	*Project 1*	*Project 2*	*Project 3*	*Project 4*
0	−100,000	−1,000,000	−100,000	−120,000
1	10,000	200,000	30,000	30,000
2	10,000	200,000	30,000	30,000
3	10,000	200,000	30,000	30,000
4	20,000	200,000	30,000	30,000
5	100,000	300,000	30,000	75,000
Net profit	50,000	100,000	50,000	75,000

Cash flows take place at the end of each year. The year 0 figure represents the initial investment made at the start of the project.

Moreover, the simple net profit takes no account of the timing of the cash flows. Projects 1 and 3 each have a net profit of £50,000 and therefore, according to this selection criterion, would be equally preferable. The bulk of the income occurs late in the life of project 1, whereas project 3 returns a steady income throughout

its life. Having to wait for a return has the disadvantage that the investment must be funded for longer. Add to that the fact that, other things being equal, estimates in the more distant future are less reliable that short-term estimates and we can see that the two projects are not equally preferable.

Payback period

The *payback period* is the time taken to break even or pay back the initial investment. Normally, the project with the shortest payback period will be chosen on the basis that an organization will wish to minimize the time that a project is 'in debt'.

Exercise 3.3

Consider the four project cash flows given in Table 3.2 and calculate the payback period for each of them.

The advantage of the payback period is that it is simple to calculate and is not particularly sensitive to small forecasting errors. Its disadvantage as a selection technique is that it ignores the overall profitability of the project – in fact, it totally ignores any income (or expenditure) once the project has broken even. Thus the fact that projects 2 and 4 are, overall, more profitable than project 3 is ignored.

Return on investment

The *return on investment* (ROI), also known as the *accounting rate of return* (ARR), provides a way of comparing the net profitability to the investment required. There are some variations on the formula used to calculate the return on investment but a straightforward common version is

$$\text{ROI} = \frac{\text{average annual profit}}{\text{total investment}} \times 100$$

Exercise 3.4

Calculating the ROI for project 1, the net profit is £50,000 and the total investment is £100,000. The return on investment is therefore calculated as

$$\begin{aligned}\text{ROI} &= \frac{\text{average annual profit}}{\text{total investment}} \times 100 \\ &= \frac{10{,}000}{100{,}000} \times 100 = 10\%\end{aligned}$$

Calculate the ROI for each of the other projects shown in Table 3.2 and decide which, on the basis of this criterion, is the most worthwhile.

The return on investment provides a simple, easy to calculate measure of return on capital and is therefore quite popular. Unfortunately it suffers from two severe disadvantages. Like the net profitability, it takes no account of the timing of the cash flows. More importantly, it is tempting to compare the rate of return with current interest rates. However, this rate of return bears no relationship to the interest rates offered or charged by banks (or any other normal interest rate) since it takes no account of the timing of the cash flows or of the compounding of interest. It is therefore, potentially, very misleading.

Net present value

Net present value (NPV) and internal rate of return (IRR) are collectively known as discounted cash flow (DCF) techniques.

The calculation of *net present value* is a project evaluation technique that takes into account the profitability of a project and the timing of the cash flows that are produced. It does so by discounting future cash flows by a percentage known as the discount rate. This is based on the view that receiving £100 today is better than having to wait until next year to receive it, because the £100 next year is worth less than £100 now. We could, for example, invest the £100 in a bank today and have £100 plus the interest in a year's time. If we say that the present value of £100 in a year's time is £91, we mean that £100 in a year's time is the equivalent of £91 now.

Note that this example uses approximate figures – when you have finished reading this section you should be able to calculate the exact figures yourself.

The equivalence of £91 now and £100 in a year's time means we are discounting the future income by approximately 10% – that is, we would need an extra 10% to make it worthwhile waiting for a year. An alternative way of considering the equivalence of the two is to consider that, if we received £91 now and invested for a year at an annual interest rate of 10%, it would be worth £100 in a year's time. The annual rate by which we discount future earnings is known as the *discount rate* – 10% in the above example.

Similarly, £100 received in 2 years' time would have a present value of approximately £83 – in other words, £83 invested at an interest rate of 10% would yield approximately £100 in 2 years' time.

The present value of any future cash flow may be obtained by applying the following formula

$$\text{present value} = \frac{\text{value in year } t}{(1+r)^t}$$

where r is the discount rate, expressed as a decimal value and t is the number of years into the future that the cash flow occurs.

Alternatively, and rather more easily, the present value of a cash flow may be calculated by multiplying the cash flow by the appropriate discount factor. A small table of discount factors is given in Table 3.3.

The NPV for a project is obtained by discounting each cash flow (both negative and positive) and summing the discounted values. It is normally assumed that any initial investment takes place immediately (indicated as year 0) and is not discounted. Later cash flows are normally assumed to take place at the end of each year and are discounted by the appropriate amount.

Table 3.3 *Table of NPV discount factors*

	Discount rate (%)					
Year	*5*	*6*	*8*	*10*	*12*	*15*
1	0.9524	0.9434	0.9259	0.9091	0.8929	0.8696
2	0.9070	0.8900	0.8573	0.8264	0.7972	0.7561
3	0.8638	0.8396	0.7938	0.7513	0.7118	0.6575
4	0.8227	0.7921	0.7350	0.6830	0.6355	0.5718
5	0.7835	0.7473	0.6806	0.6209	0.5674	0.4972
6	0.7462	0.7050	0.6302	0.5645	0.5066	0.4323
7	0.7107	0.6651	0.5835	0.5132	0.4523	0.3759
8	0.6768	0.6274	0.5403	0.4665	0.4039	0.3269
9	0.6446	0.5919	0.5002	0.4241	0.3606	0.2843
10	0.6139	0.5584	0.4632	0.3855	0.3220	0.2472
15	0.4810	0.4173	0.3152	0.2394	0.1827	0.1229
20	0.3769	0.3118	0.2145	0.1486	0.1037	0.0611
25	0.2953	0.2330	0.1460	0.0923	0.0588	0.0304

More extensive or detailed tables may be constructed using the formula

$$\text{discount factor} = \frac{1}{(1+r)^t}$$

for various values of r (the discount rate) and t (the number of years from now)

Exercise 3.5

Assuming a 10% discount rate, the NPV for project 1 (Table 3.2) would be calculated as in Table 3.4. The net present value for Project 1, using a 10% discount rate is therefore £618. Using a 10% discount rate, calculate the net present values for projects 2, 3 and 4 and decide which, on the basis of this, is the most beneficial to pursue.

Table 3.4 *Applying the discount factors to project 1*

Year	*Project 1 cash flow (£)*	*Discount factor @ 10%*	*Discounted cash flow (£)*
0	−100,000	1.0000	−100,000
1	10,000	0.9091	9,091
2	10,000	0.8264	8,264
3	10,000	0.7513	7,513
4	20,000	0.6830	13,660
5	100,000	0.6209	62,090
Net Profit:	£50,000		NPV: £618

It is interesting to note that the net present values for projects 1 and 3 are significantly different – even though they both yield the same net profit and both have the same return on investment. The difference in NPV reflects the fact that, with project 1, we must wait longer for the bulk of the income.

The main difficulty with NPV for deciding between projects is selecting an appropriate discount rate. Some organizations have a standard rate but, where this is not the case, then the discount rate should be chosen to reflect available interest rates (borrowing costs where the project must be funded from loans) plus some premium to reflect the fact that software projects are inherently more risky than lending money to a bank. The exact discount rate is normally less important than ensuring that the same discount rate is used for all projects being compared. However, it is important to check that the ranking of projects is not sensitive to small changes in the discount rate – have a look at the following exercise.

Exercise 3.6

Calculate the net present value for each of the projects A, B and C shown in Table 3.5 using each of the discount rates 8%, 10% and 12%.

For each of the discount rates, decide which is the best project. What can you conclude from these results?

Alternatively, the discount rate can be thought of as a target rate of return. If, for example, we set a target rate of return of 15% we would reject any project that did not display a positive net present value using a 15% discount rate. Any project that displayed a positive NPV would be considered for selection – perhaps by using an additional set of criteria where candidate projects were competing for resources.

Table 3.5 *Three estimated project cash flows*

Year	*Project A (£)*	*Project B (£)*	*Project C (£)*
0	– 8,000	– 8,000	– 10,000
1	4,000	1,000	2,000
2	4,000	2,000	2,000
3	2,000	4,000	6,000
4	1,000	3,000	2,000
5	500	9,000	2,000
6	500	–6,000	2,000
Net Profit	4,000	5,000	6,000

Internal rate of return

One disadvantage of NPV as a measure of profitability is that, although it may be used to compare projects, it might not be directly comparable with earnings from other investments or the costs of borrowing capital. Such costs are usually quoted

as a percentage interest rate. The internal rate of return (IRR) attempts to provide a profitability measure as a percentage return that is directly comparable with interest rates. Thus, a project that showed an estimated IRR of 10% would be worthwhile if the capital could be borrowed for less than 10% or if the capital could not be invested elsewhere for a return greater than 10%.

The IRR is calculated as that percentage discount rate that would produce an NPV of zero. It is most easily calculated using a spreadsheet or other computer program that provides functions for calculating the IRR. Microsoft Excel and Lotus, for example, both provide IRR functions which, provided with an initial guess or seed value (which may be zero), will search for and return an IRR.

Manually, it must be calculated by trial-and-error or estimated using the technique illustrated in Figure 3.3. This technique consists of guessing two values

The IRR may be estimated by plotting a series of guesses: For a particular project, a discount rate of 8% gives a positive NPV of £7,898; a discount rate of 12% gives a negative NPV of −£5,829. The IRR is therefore somewhere between these two values. Plotting the two values on a chart and joining the points with a straight line suggests that the IRR is about 10.25%. The true IRR (calculated with a spreadsheet) is 10.167%.

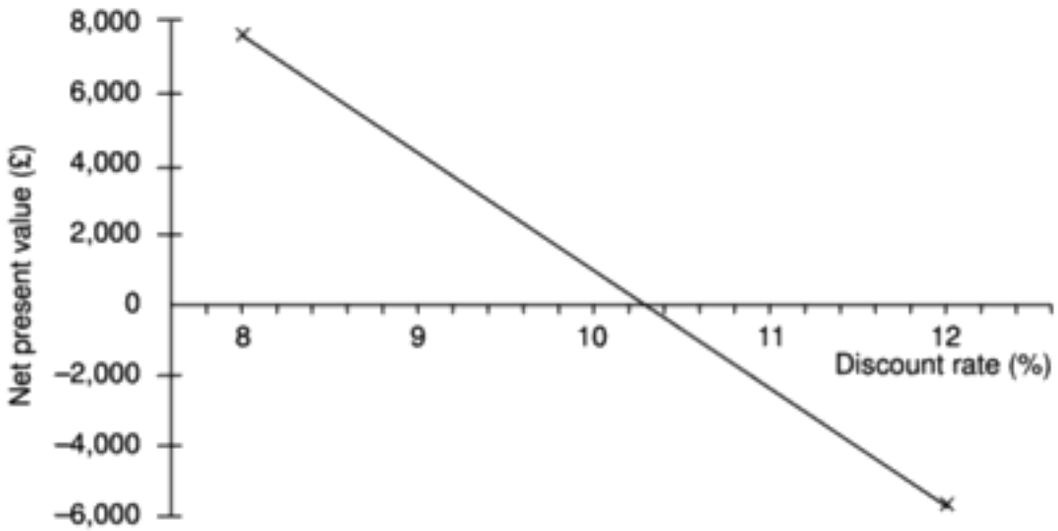


Figure 3.3 *Estimating the internal rate of return for project 1.*

(one either side of the true value) and using the resulting NPVs (one of which must be positive and the other negative) to estimate the correct value. Note that this technique will provide only an approximate value but, in many cases that can be sufficient to dismiss a project that has a small IRR or indicate that it is worth making a more precise evaluation.

The internal rate of return is a convenient and useful measure of the value of a project in that it is a single percentage figure that may be directly compared with rates of return on other projects or interest rates quoted elsewhere.

Table 3.6 illustrates the way in which a project with an IRR of 10% may be directly compared with other interest rates. The cash flow for the project is shown in column (a). Columns (b) to (e) show that if we were to invest £100,000 now at an annual interest rate of 10% in, say, a bank, we could withdraw the same amounts as we would earn from the project at the end of each year, column (e), and we would be left with a net balance of zero at the end. In other words, investing in a project that has an IRR of 10% can produce exactly the same cash flow as lending the money to a bank at a 10% interest rate. We can therefore reason

that a project with an IRR greater than current interest rates will provide a better rate of return than lending the investment to a bank. We can also say that it will be worth borrowing to finance the project if it has an IRR greater than the interest rate charged on the loan.

Table 3.6 *A project cash flow treated as an investment at 10%*

		Equivalent investment at 10%			
Year	*(a) Project cash flow forecast (£)*	*(b) Capital at start of year (£)*	*(c) Interest during year (£)*	*(d) Capital at end of year (£)*	*(e) End of year withdrawal (£)*
0	−100,000	—	—	—	—
1	10,000	100,000	10,000	110,000	10,000
2	10,000	100,000	10,000	110,000	10,000
3	10,000	100,000	10,000	110,000	10,000
4	20,000	100,000	10,000	110,000	20,000
5	99,000	90,000	9,000	99,000	99,000
6		0	0	0	0

£100,000 invested at 10% may be used to generate the cash flows shown. At the end of the 5-year period the capital and the interest payments will be entirely consumed leaving a net balance of zero.

One deficiency of the IRR is that it does not indicate the absolute size of the return. A project with an NPV of £100,000 and an IRR of 15% can be more attractive than one with an NPV of £10,000 and an IRR of 18% – the return on capital is lower but the net benefits greater.

An often quoted objection to the internal rate of return is that, under certain conditions, it is possible to find more than one rate that will produce a zero NPV. This is not a valid objection since, if there are multiple solutions, it is always appropriate to take the lowest value and ignore the others. Spreadsheets will normally always return the lowest value if provided with zero as a seed value.

NPV and IRR are not, however, a complete answer to economic project evaluation.

- A total evaluation must also take into account the problems of funding the cash flows – will we, for example, be able to repay the interest on any borrowed money and pay development staff salaries at the appropriate time?
- While a project's IRR might indicate a profitable project, future earnings from a project might be far less reliable than earnings from, say, investing with a bank. To take account of the risk inherent in investing in a project, we might require that a project earn a 'risk premium' (that is, it must earn, say, at least 15% more than current interest rates) or we might undertake a more detailed risk analysis as described in the following sections of this chapter.
- We must also consider any one project within the financial and economic framework of the organization as a whole – if we fund this one, will we also be able to fund other worthy projects?

3.7 Risk evaluation

There is a risk that software might exceed the original specification and that a project will be completed early and under budget. That is not a risk that need concern us.

Every project involves risk of some form. When assessing and planning a project, we are concerned with the risk of the project's not meeting its objectives. In Chapter 8 we shall discuss ways of analysing and minimizing risk during the development of a software system. In this chapter, we are concerned with taking risk into account when deciding whether to proceed with a proposed project.

Risk identification and ranking

In any project evaluation we should attempt to identify the risks and quantify their potential effects. One common approach to risk analysis is to construct a project risk matrix utilizing a checklist of possible risks and to classify each risk according to its relative importance and likelihood. Note that the importance and likelihood need to be separately assessed – we might be less concerned with something that, although serious, is very unlikely to occur than with something less serious that is almost certain. Table 3.7 illustrates a basic project risk matrix listing some of the risks that might be considered for a project, with their importance and likelihood classified as high (H), medium (M), low (L) or exceedingly unlikely (—). So that projects may be compared the list of risks must be the same for each project being assessed. It is likely, in reality, that it would be somewhat longer than shown and more precisely defined.

The project risk matrix may be used as a way of evaluating projects (those with high risks being less favoured) or as a means of identifying and ranking the risks for a specific project. In Chapter 7 we shall consider a method for scoring the importance and likelihood of risks that may be used in conjunction with the project risk matrix to score and rank projects.

Risk and net present value

Where a project is relatively risky it is common practice to use a higher discount rate to calculate net present value. This addition or risk premium, might, for example, be an additional 2% for a reasonably safe project or 5% for a fairly risky one. Projects may be categorized as high, medium or low risk using a scoring method and risk premiums designated for each category. The premiums, even if arbitrary, provide a consistent method of taking risk into account.

Cost–benefit analysis

A rather more sophisticated approach to the evaluation of risk is to consider each possible outcome and estimate the probability of its occurring and the corresponding value of the outcome. Rather than a single cash flow forecast for a project, we will then have a set of cash flow forecasts, each with an associated probability of occurring. The value of the project is then obtained by summing the cost or benefit for each possible outcome weighted by its corresponding probability. Exercise 3.7 illustrates how this may be done.

Table 3.7 *A fragment of a basic project risk matrix*

Risk	*Importance*	*Likelihood*
Software never completed or delivered	H	—
Project cancelled after design stage	H	—
Software delivered late	M	M
Development budget exceeded ≤ 20%	L	M
Development budget exceeded > 20%	M	L
Maintenance costs higher than estimated	L	L
Response time targets not met	L	H

Exercise 3.7

BuyRight, a software house, is considering developing a payroll application for use in academic institutions and is currently engaged in a cost-benefit analysis. Study of the market has shown that, if they can target it efficiently and no competing products become available, they will obtain a high level of sales generating an annual income of £800,000. They estimate that there is a 1 in 10 chance of this happening. However, a competitor might launch a competing application before their own launch date and then sales might generate only £100,000 per year. They estimate that there is a 30% chance of this happening. The most likely outcome, they believe, is somewhere in between these two extremes – they will gain a market lead by launching before any competing product becomes available and achieve an annual income of £650,000. BuyRight have therefore calculated their expected sales income as in Table 3.8.

Total development costs are estimated at £750,000 and sales are expected to be maintained at a reasonably constant level for at least four years. Annual costs of marketing and product maintenance are estimated at £200,000, irrespective of the market share gained. Would you advise them to go ahead with the project?

This approach is frequently used in the evaluation of large projects such as the building of new motorways, where variables such as future traffic volumes, and hence the total benefit of shorter journey times, are subject to uncertainty. The technique does, of course, rely on our being able to assign probabilities of occurrence to each scenario and, without extensive study, this can be difficult.

When used to evaluate a single project, the cost–benefit approach, by 'averaging out' the effects of the different scenarios, does not take account an organization's reluctance to risk damaging outcomes. Because of this, where overall profitability is the primary concern, it is often considered more appropriate for the evaluation of a portfolio of projects.

Risk profile analysis

An approach which attempts to overcome some of the objections to cost–benefit averaging is the construction of risk profiles using sensitivity analysis.

Table 3.8 *BuyRight's income forecasts*

Sales	*Annual sales income (£)* i	*Probability* p	*Expected Value (£)* $i \times p$
High	800,000	0.1	80,000
Medium	650,000	0.6	390,000
Low	100,000	0.3	30,000
Expected Income			500,000

This involves varying each of the parameters that affect the project's cost or benefits to ascertain how sensitive the project's profitability is to each factor. We might, for example, vary one of our original estimates by plus or minus 5% and recalculate the expected costs and benefits for the project. By repeating this exercise for each of our estimates in turn we can evaluate the sensitivity of the project to each factor.

By studying the results of a sensitivity analysis we can identify those factors that are most important to the success of the project. We then need to decide whether we can exercise greater control over them or otherwise mitigate their effects. If neither is the case, then we must live with the risk or abandon the project.

For an explanation of the Monte Carlo technique see any textbook on operational research.

Sensitivity analysis demands that we vary each factor one at a time. It does not easily allow us to consider the effects of combinations of circumstances, neither does it evaluate the chances of a particular outcome occurring. In order to do this we need to use a more sophisticated tool such as Monte Carlo simulation. There are a number of risk analysis applications available (such as *@Risk* from Palisade) that use Monte Carlo simulation and produce risk profiles of the type shown in Figure 3.4.

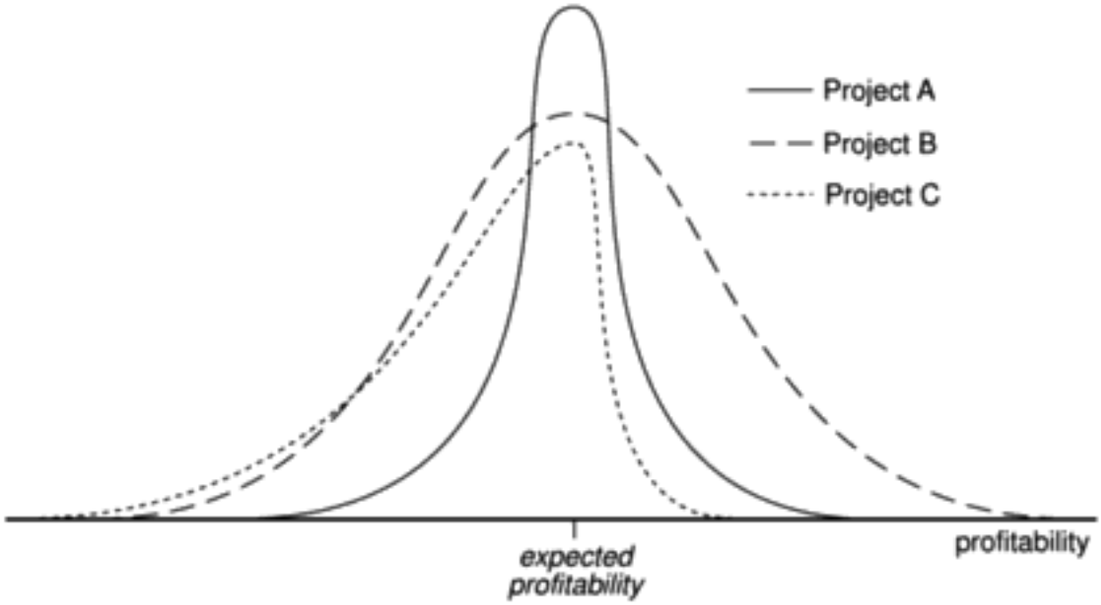
Projects may be compared as in Figure 3.4, which compares three projects with the same expected profitability. Project A is unlikely to depart far from this expected value compared to project B, which exhibits a larger variance. Both of these have symmetric profiles, which contrast with project C. Project C has a skewed distribution, which indicates that although it is unlikely ever to be much more profitable than expected, it is quite likely to be far worse.

Using decision trees

The approaches to risk analysis discussed previously rather assume that we are passive bystanders allowing nature to take its own course – the best we can do is to reject over-risky projects or choose those with the best risk profile. There are many situations, however, where we can evaluate whether a risk is important and, if it is, indicate a suitable course of action.

Many such decisions will limit or affect future options and, at any point, it is important to be able to see into the future to assess how a decision will affect the future profitability of the project.

Prior to giving Amanda the job of extending their invoicing system, IOE must consider the alternative of completely replacing the existing system – which they



All three projects have the same expected profitability.
The profitability of project A is unlikely to depart greatly from its expected value (indicated by the vertical axis) compared to the likely variations for project B. Project A is therefore less risky than project B.

Figure 3.4 *A risk analysis profile.*

will have to do at some point in the future. The decision largely rests upon the rate at which their equipment maintenance business expands – if their market share significantly increases (which they believe will happen if rumours of a competitor's imminent bankruptcy are fulfilled) the existing system might need to be replaced within 2 years. Not replacing the system in time could be an expensive option as it could lead to lost revenue if they cannot cope with the increase in invoicing demand. Replacing it immediately will, however, be expensive as it will mean deferring other projects that have already been scheduled.

They have calculated that extending the existing system will have an NPV of £57,000, although if the market expands significantly, this will be turned into a loss with an NPV of –£100,000 due to lost revenue. If the market does expand, replacing the system now has an NPV of £250,000 due to the benefits of being able to handle increased sales and other benefits such as improved management information. If sales do not increase, however, the benefits will be severely reduced and the project will suffer a loss with an NPV of –£50,000.

The company estimate the likelihood of the market increasing significantly at 20% – and, hence, the probability that it will not increase as 80%. This scenario can be represented as a tree structure as shown in Figure 3.5.

The analysis of a decision tree consists of evaluating the expected benefit of taking each path from a decision point (denoted by D in Figure 3.5). The expected value of each path is the sum of the value of each possible outcome multiplied by its probability of occurrence. The expected value of extending the system is therefore £40,000 ($75{,}000 \times 0.8 - 100{,}000 \times 0.2$) and the expected value of replacing the system £10,000 ($250{,}000 \times 0.2 - 50{,}000 \times 0.8$). IOE should therefore choose the option of extending the existing system.

This example illustrates the use of a decision tree to evaluate a simple decision at the start of a project. One of the great advantages of using decision trees to

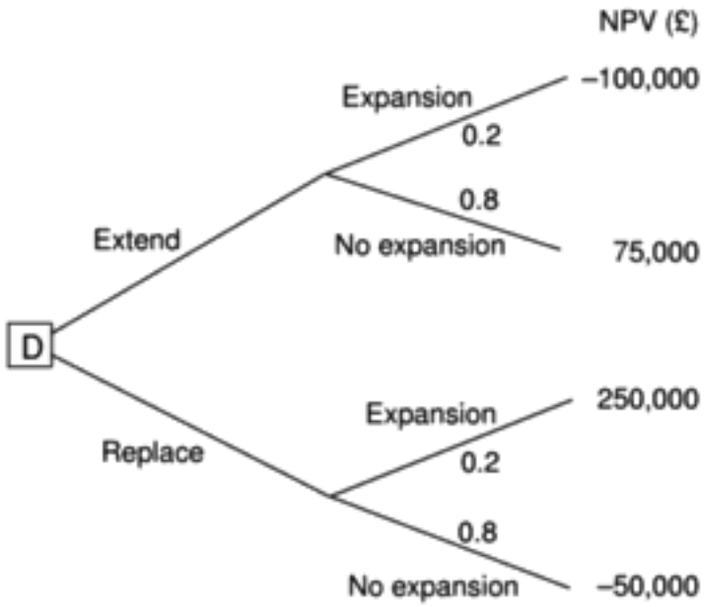


Figure 3.5 *A decision tree.*

model and analyse problems is the ease with which they can be extended. Figure 3.6 illustrates an extended version of Amanda's decision tree, which includes the possibility of a later decision should they decide to extend the system and then find there is an early market expansion.

The net present values shown in italic are those identified in Amanda's original decision tree shown in Figure 3.5.

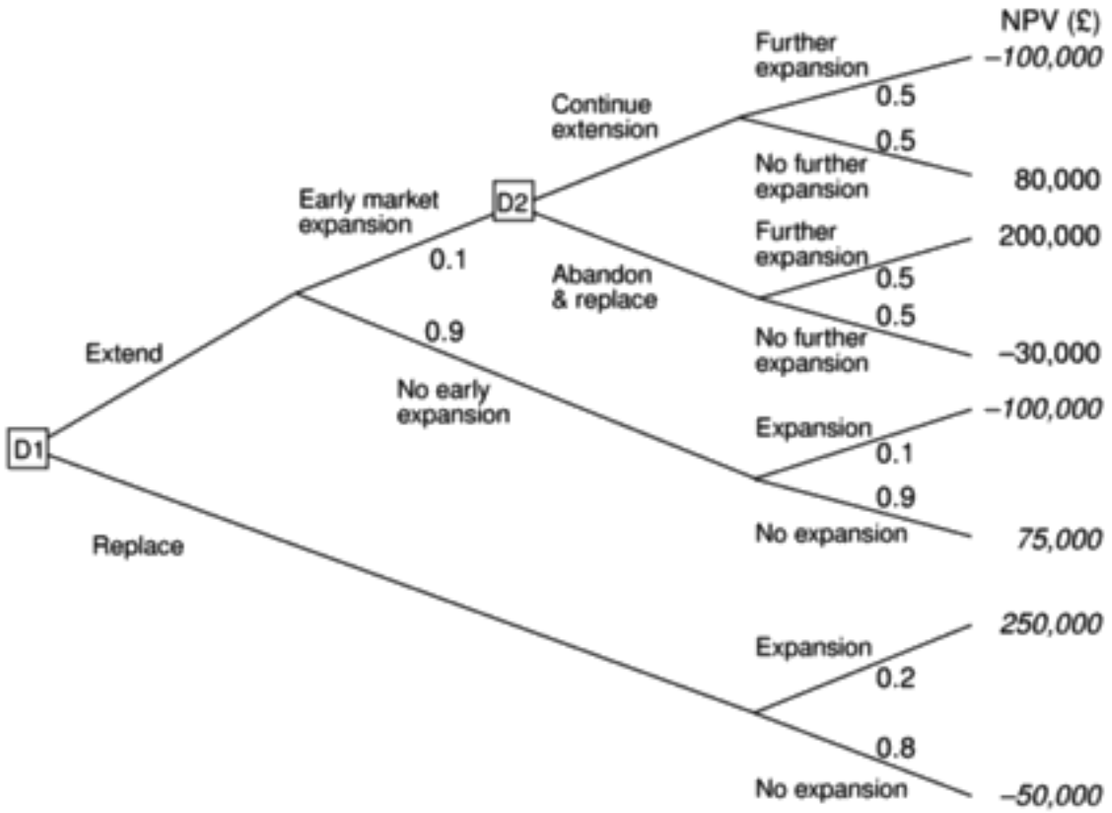


Figure 3.6 *An extension to Amanda's decision tree.*

3.8 Conclusion

Some of the key points in this chapter are:

- projects must be evaluated on strategic, technical and economic grounds;
- economic assessment involves the identification of all costs and income over the lifetime of the system, including its development and operation and checking that the total value of benefits exceeds total expenditure;
- money received in the future is worth less than the same amount of money in hand now, which may be invested to earn interest;
- the uncertainty surrounding estimates of future returns lowers their real value measured now;
- discounted cash flow techniques may be used to evaluate the present value of future cash flows taking account of interest rates and uncertainty;
- cost–benefit analysis techniques and decision trees provide tools for evaluating expected outcomes and choosing between alternative strategies.

3.9 Further exercises

1. Identify the major risks that could affect the success of the Brightmouth College Payroll project and try to rank them in order of importance.
2. Working in a group of three or four, imagine that you are about to embark upon a programming assignment as part of the assessed work for your course. Draw up a list of the risks that might affect the assignment outcome. Individually classify the importance and likelihood of each of those risk as high, medium or low. When you have done this compare your results and try to come up with an agreed project risk matrix.
3. Explain why discounted cash flow techniques provide better criteria for project selection than net profit or return on investment.
4. Consider the decision tree shown in Figure 3.6 and decide, given the additional possibilities, which option(s) IOE should choose.

Chapter 4

Selection of an appropriate project approach

OBJECTIVES

When you have completed this chapter you will be able to:

- take account of the characteristics of the system to be developed when planning a project;
- select an appropriate process model;
- make best use of the 'waterfall' process model where appropriate;
- reduce risks by the creation of appropriate prototypes;
- reduce other risks by implementing the project in increments.

4.1 Introduction

The development of software in-house suggests that the project has certain characteristics:

- the project team and the users belong to the same organization;
- the projects being considered slot into a portfolio of existing computer-based systems;
- the methodologies and technologies to be used are not selected by the project manager, but are dictated by local standards.

However, where a series of development projects is being carried out by a software house for different external customers, the methodologies and technologies to be used will have to be decided for each individual project. This decision-making process has been called 'technical planning' by some, although here we use the term 'project analysis'. Even where development is in-house, it is important to spend some time looking for any characteristics of the new project that might make us take a different approach from that used on previous projects. It is this analysis that is the subject of this chapter.

Martyn Ould in *Strategies for Software Engineering: the Management of Risk & Quality*, John Wiley & Sons, 1990, describes how technical planning was done at the software house, Praxis.

The relevant part of the Step Wise approach is Step 3: Analyse project characteristics. The selection of a particular process model will add new products to the Project Breakdown Structure or new activities to the activity network. This will create outputs for Step 4: Identify the products and activities of the project (see Figure 4.1).

In the remainder of this chapter we will firstly look at how the characteristics of a project will influence the approach to the planning of a project. We will then look at some of the most common *process models*, namely the waterfall approach, prototyping and incremental delivery.

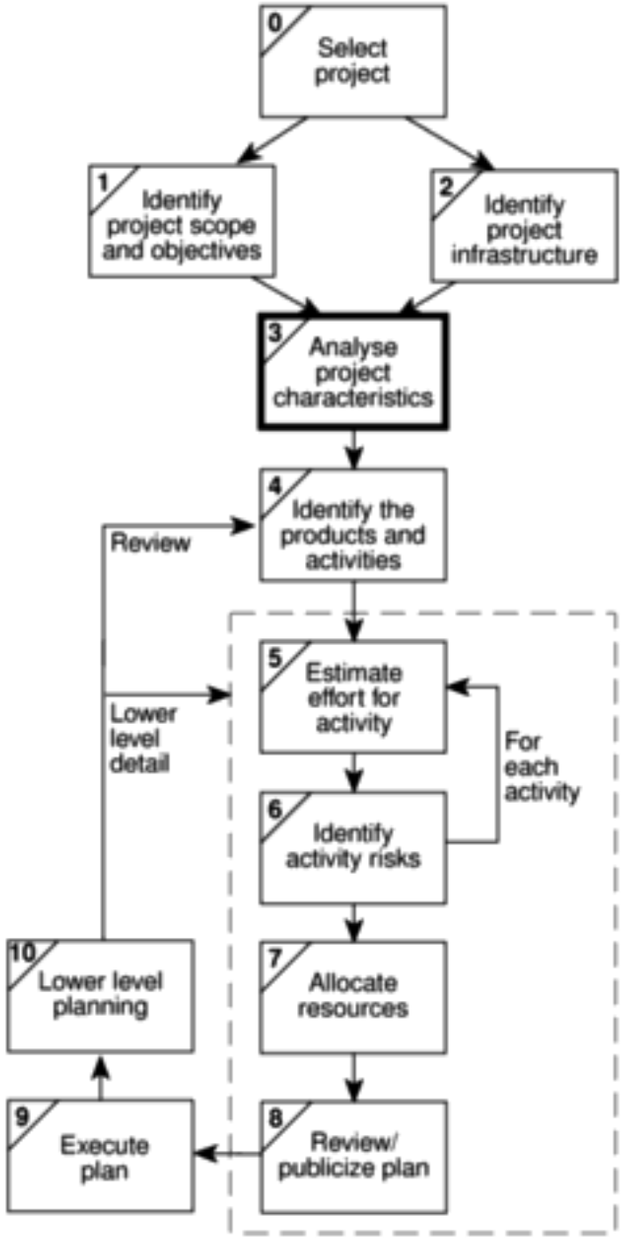


Figure 4.1 *Project analysis is the subject of Step 3.*

4.2 Choosing technologies

An outcome of project analysis will be the selection of the most appropriate methodologies and technologies. Methodologies include techniques like the various flavours of object-oriented (OO) development, SSADM and JSP (Jackson Structured Programming) while technologies might include an appropriate application-building environment, or the use of knowledge-based system tools.

As well as the products and activities, the chosen technology will influence the following aspects of a project:

- the training requirements for development staff;
- the types of staff to be recruited;
- the development environment – both hardware and software;
- system maintenance arrangements.

We are now going to describe some of the steps of project analysis.

Identify project as either objectives-driven or product-driven

You will recall from Chapter 1 that we distinguished between objectives-driven and product-driven projects. Very often a product-driven project will have been preceded by an objectives-driven project which chose the general software solution that is to be implemented.

There will be cases where things are so vague that even the objectives of the project are uncertain or are the subject of disagreement. People may be experiencing a lot of problems but no-one knows exactly what the solution to the problems might be. It could be that the IT specialists can provide help in some places but assistance from other specialisms is needed in others. In these kinds of situation a *soft systems* approach might need to be considered.

The soft systems approach is described in Checkland, P. and Scholes, J., *Soft systems methodology in action*, John Wiley and Sons, 1990.

Analyse other project characteristics

The sorts of question that would need to be asked include the following.

- **Is a data orientated or a control orientated system to be implemented?** 'Data orientated' systems generally mean information systems that will have a considerable database. 'Control orientated' systems refer to embedded control systems. These days it is not uncommon to have systems with components of both types.

We first introduced the difference between information systems and embedded systems in Chapter 1.

- **Will the software that is to be produced be a general package or application specific?** An example of a general package would be a spreadsheet or a word processing package. An application specific package could be, for example, an airline seat reservation system.
- **Is the system to be implemented of a particular type for which specific tools have been developed?** For example:

  - *does it involve concurrent processing?* – if so the use of techniques appropriate to the analysis and design of such systems would be considered;
  - *will the system to be created be knowledge-based?* – expert systems have a set of rules which result in some 'expert advice' when applied to a problem domain (sets of methods and tools have been developed to assist in the creation of such systems); or
  - will the system to be produced make heavy use of computer graphics?
- **Is the system to be created safety-critical?** For instance, could a malfunction in the system endanger human life?
- **What is the nature of the hardware/software environment in which the system will operate?** It might be that the environment in which the final software will operate is different from that in which it will be developed. Embedded software may be developed on a large development machine that has lots of supporting software tools in the way of compilers, debuggers, static analysers and so on, but might then be down-loaded to a small processor in the larger engineered product. A system destined for a personal computer will need a different approach to one destined for a main-frame or a client-server environment.

Exercise 4.1

How would you categorize each of the following systems according to the classification above?

(a) a payroll system
(b) a system to control a bottling plant
(c) a system that holds details of the plans of plant used by a water company to supply water to consumers
(d) a software application to support project managers
(e) a system used by lawyers to get hold of case law relating to company taxation.

Identify high level project risks

Chapter 3 has already touched on some aspects of risk, which are developed further in Chapter 8.

When we first embark on a project we might be expected to work out elaborate plans even though we are at least partially ignorant of many important factors that will affect the project. For example, until we do a detailed investigation of the users' requirements we will not be able to estimate how much effort will be needed to build a system to meet those requirements. The greater the uncertainties at the beginning of the project, the greater the risk that the project will be unsuccessful. Once we recognize a particular area of uncertainty we can, however, take steps to reduce its uncertainty.

One suggestion is that uncertainty can be associated with the *products*, *processes*, or *resources* associated with the project.

Product uncertainty Here we ask how well the requirements are understood. It might be that the users themselves are uncertain about what a proposed information system is to do. The government, say, might introduce a new form of taxation but the way this is going to operate in detail will not be known until a certain amount of case law has been built up. Some environments can change so quickly that what was a precise and valid statement of requirements rapidly becomes out of date.

Process uncertainty It might be that the project under consideration is the first where an organization has tried to use a method, such as SSADM or OMT, that is new to them. Perhaps a new application building tool is being used. Any change in the way that the systems are developed is going to introduce uncertainty.

OMT is an object-oriented design approach.

Resource uncertainty The main area of uncertainty here will almost surely be the availability of staff of the right ability and experience. A major influence on the degree of uncertainty in a project will be the sheer size of a project. The larger the number of resources needed or the longer the duration of the project, the more inherently risky it is likely to be.

Euromethod, which is reviewed briefly in Appendix C, distinguishes between factors that increase *uncertainty*, for example, continually changing requirements, and those that increase *complexity*, for example, software size. This is because it suggests different strategies to deal with the two distinct types of risk.

Exercise 4.2

At IOE, Amanda has identified possible staff resistance as a risk to the maintenance group accounts project. Would you classify this as a product, process or resource risk? Perhaps it does not fit into any of these categories and some other is needed.

Brigette at Brightmouth College identified the possibility that no suitable payroll package would be available on the market as a risk. What other risks might be inherent in the Brightmouth College payroll project?

Take into account user requirements concerning implementation

A user organization lays down standards that have to be adopted by any contractor providing software for them. For example the UK Civil Service favours the SSADM standard where information systems are being developed.

It is common for organizations to specify that suppliers of software have BS EN ISO 9001:1994 or TickIT accreditation. This will affect the way projects are conducted.

Chapter 12, Software quality, discusses BS EN ISO 9001 and TickIT.

Select general life cycle approach

Control systems A real-time system will have to be implemented using an appropriate methodology, for example, Mascot. Real-time systems that employ concurrent processing will use techniques such as Petri nets.

Information systems Similarly, an information system will need a methodology, such as SSADM or Information Engineering, that matches that type of environment. SSADM will be especially appropriate where the project will employ a large number of development staff whose work will need to be co-ordinated: the method lays down in detail what needs to be done and what products need to be created at each step. Team members would therefore know exactly what is expected of them.

General applications Where the software to be produced is for the general market rather than for a specific application and user, then a methodology such as SSADM would have to be thought about very carefully. This is because the framers of the method make the assumption that a specific user exists. Some parts in the method also assume that there is an existing system that can be analysed to yield the logical features of the new, computer-based, system.

Specialized techniques These have been invented to expedite the development of, for example, *knowledge-based systems* where there are a number of specialized tools and logic based programming languages that can be used to implement this type of system. Similarly a number of specialized techniques and tools have been developed to assist in the development of *graphics-based systems*.

Hardware environment The environment in which the system is to operate can put constraints on the way it is to be implemented. For instance, the need for a fast response time or for the software to take up only a small part of computer memory may mean that only low-level programming languages can be used – particularly in real-time and embedded systems.

Safety-critical systems Where safety and reliability are of the essence, it might be possible to justify the additional expense of a formal specification using a notation such as Z or VDM. Really critical systems call for expensive measures such as having independent teams develop parallel systems with the same functionality. The parallel systems can then run concurrently when the application is in operation so that the results of each of the parallel systems can be cross-checked.

The implications of prototyping and the incremental approach are explored later in the chapter.

Imprecise requirements Uncertainties or a *novel hardware/software platform* may mean that a *prototyping* approach should be considered. If the environment in which the system is to be implemented is a rapidly changing one, then serious consideration would need to be given to *incremental delivery*. If the users have *uncertain objectives* in connection with the project, then a *soft systems* approach might be desirable.

Exercise 4.3

Bearing in mind the discussion above, what, in broad outline, is the most suitable approach for each of the following?

(a) a system that calculates the amount of a drug that should be administered to a patient who has a particular complaint;

(b) a system to administer a student loans scheme;
(c) a system to control trains on a railway network.

4.3 Technical plan contents list

The analysis described above will produce a number of practical requirements that will be fed into the next stage of the planning process. These requirements might add activities to the project and might involve the acquisition of items of software or hardware or the adoption of particular methodologies which might require staff training. As these recommendations imply certain costs, they should be recorded formally.

A preliminary version of this technical plan can be produced by a software house to help in the preparation of the bid for a contract. In some cases, it may actually be shown to the potential customer in order to show the basis for the bid price and generally to impress the customer with the soundness of the approach the software house intends to adopt.

The technical plan is likely to have the following contents.

1. Introduction and summary of constraints:
   (a) character of the system to be developed;
   (b) risks and uncertainties of the project;
   (c) user requirements concerning implementation.
2. Recommended approach:
   (a) selected methodology or process model;
   (b) development methods;
   (c) required software tools;
   (d) target hardware/software environment.
3. Implementation:
   (a) required development environment;
   (b) required maintenance environment;
   (c) required training.
4. Implications:
   (a) project products and activities – these will have an effect on the schedule duration and overall project effort;
   (b) financial – this report will be used to produce costings.

4.4 Choice of process models

The word ‘process’ is sometimes used to emphasize the idea of a system *in action*. In order to achieve an outcome, the system will have to execute one or more activities: this is its process. This idea can be applied to the development of computer-based systems where a number of interrelated activities have to be

undertaken to create a final product. These activities can be organized in different ways and we can call these *process models*.

A major part of the planning will be the choosing of the development methods to be used and the slotting of these into an overall process model.

The planner needs not only to select methods but also to specify how the method is to be applied. With methods such as SSADM, there is a considerable degree of choice about how it is to be applied: not all parts of SSADM are compulsory. Many student projects have the rather basic failing that at the planning stage they claim that, say, SSADM is to be used: in the event, all that is produced are a few SSADM fragments such as a top level data flow diagram and a preliminary logical data structure diagram. If this is all the particular project requires, it should be stated at the outset.

4.5 Structured methods

Although some 'object-oriented' specialists may object(!), we include the OO approach as a structured method – after all, we hope it is not unstructured. Structured methods are made up of sets of steps and rules, which, when applied produce system products such as data flow diagrams. Each of these products is carefully documented. Such methods are often time consuming compared to more intuitive approaches and this implies some additional cost. The pay-off is such things as a less error prone and more maintainable final system. This balance of costs and benefits is more likely to be justified on a large project involving many developers and users. This is not to say that smaller projects cannot justify the use of such methods.

4.6 Rapid application development

It might be thought that users would generally welcome the more professional approach that structured methods imply. However, customers of IS/IT are concerned with getting working business applications delivered quickly and at less cost and often see structured methods as unnecessarily bureaucratic and slow. A response to this has been *rapid application development* (RAD).

The RAD approach does not preclude the use of some elements of structured methods such as Logical Data Structure diagrams but also adopts tactics such as *joint application development* (JAD) workshops. In these workshops, developers and users work together intensively for, say, three to five days and identify and agree fully documented business requirements. Often, these workshops are conducted away from the normal business and development environments in *clean rooms*, that is, special conference rooms free from outside interruption and suitably furnished with white boards and other aids to communication. This is based on the belief that communication and negotiation that might take several weeks or months via a conventional approach of formal reports and proposals and counter-proposals can be speeded up in these hothouse conditions.

Another practice associated with RAD is *time-boxing* where the scope of a project is rigidly constrained by a pre-agreed and relatively close absolute deadline. Requirements that have to be omitted in order to meet this deadline are considered in later time-boxes.

Some of these ideas will be considered further in our later discussions on prototyping and the incremental approach.

4.7 The waterfall model

This is the 'classical' model of system development. An alternative name for this model is the *one-shot* approach. As can be seen from the example in Figure 4.2, there is a sequence of activities working from top to bottom. The diagram shows some arrows pointing upwards and backwards. This indicates that a later stage might reveal the need for some extra work at an earlier stage, but this should definitely be the exception rather the rule. After all, the flow of a waterfall should be downwards with the possibility of just a little splashing back. The limited scope for iteration is in fact one of the strengths of this process model. With a large project you want to avoid having to go back and rework tasks that you thought had been completed. For a start, having to reopen what was previously thought to be a completed activity plays havoc with promised completion dates.

The first description of this approach is said to be that of H. D. Bennington in 'Production of Large Computer Programs' in 1956. This was reprinted in 1983 in *Annals of the History of Computing* 5(4).

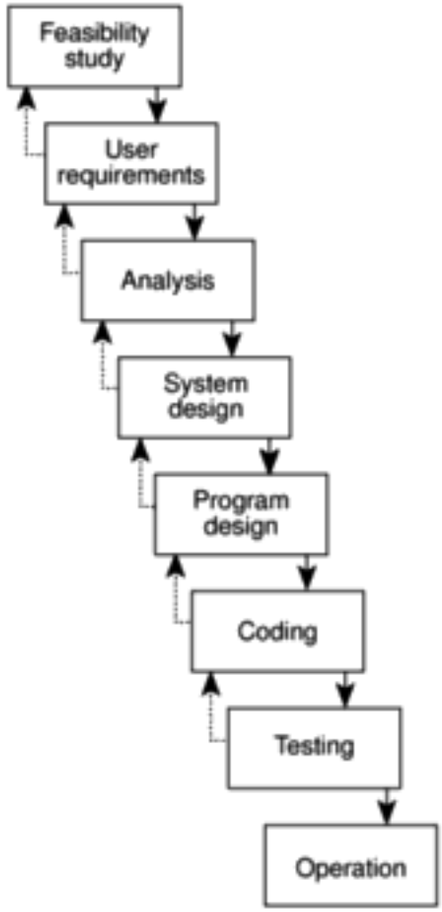


Figure 4.2 *The waterfall model.*

We contend that there is nothing intrinsically wrong with the waterfall approach, even though more recent writers have suggested different models. It is the ideal that the project manager strives for. The waterfall approach allows project completion times to be forecast with more confidence than is the case with some more iterative approaches and this allows projects to be controlled effectively. However, where there is uncertainty about how a system is to be implemented, and unfortunately there very often is, a more flexible, iterative, approach may be required.

4.8 The V-process model

Figure 4.3 gives a diagrammatic representation of this model. This is an elaboration of the waterfall model and stresses the necessity for validation activities that match the activities that create the products of the project.

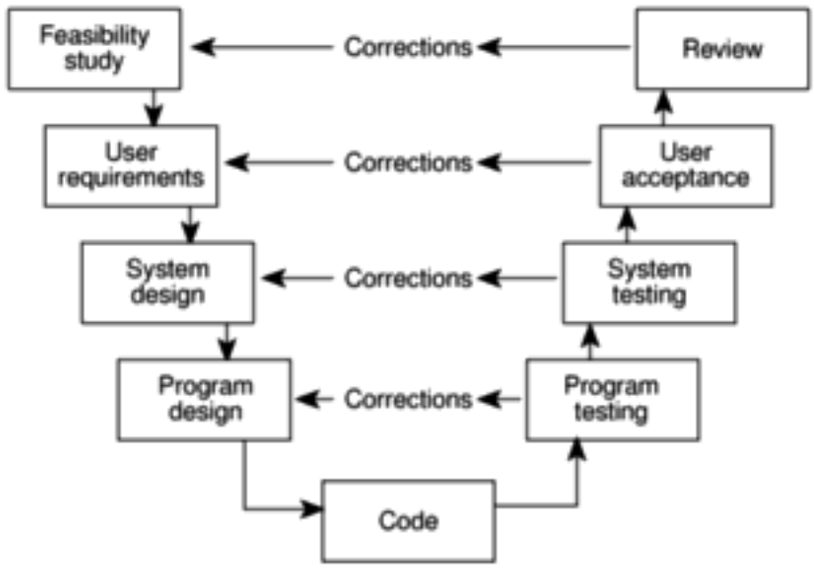


Figure 4.3 *The V-process model.*

The V-process model can be seen as expanding the testing activity in the waterfall model. Each step has a matching validation process that can, where defects are found, cause a loop back to the corresponding development stage and a reworking of the succeeding steps. Ideally, this feeding back should only occur where a discrepancy has been found between what was specified by a particular activity and what was actually implemented in the next lower activity on the descent of the V loop. For example, the system designer might have written that a calculation be carried out in a certain way. The person who structured the software that fulfilled this design might have misunderstood what was required. At system testing stage, the system designer would carry out checks that ensure that the software is doing what was specified in the design document and would discover the program designer's misreading of that document. Only corrections should be

fed back, not the system designer's second thoughts, otherwise the project would be slipping into an 'evolutionary prototyping' approach.

Exercise 4.4

Figure 4.3 shows the V-process model. The review that is held after the system has been implemented is shown as possibly feeding corrections back to the feasibility study which was probably conducted months or years before. How would this work in practice?

4.9 The spiral model

It could be argued that this is another way of looking at the basic waterfall model. In the waterfall model, there is a possible escape at the end of any of the activities in the sequence. A feasibility study might decide that the implementation of a proposed system would be beneficial. The management therefore authorize work on the detailed collection and analysis of user requirements. Some analysis, for instance the interviewing of users, might already have taken place at the feasibility stage, but a more thorough investigation is now launched. This might reveal that in fact the costs of implementing the system would be higher than originally estimated and lead managers to decide to abandon the project.

The original ideas behind the spiral model can be found in B. W. Boehm's 1988 paper 'A spiral model of software development and enhancement' in *IEEE Computer* 21(5).

A greater level of detail is considered at each stage of the project and a greater degree of confidence about the probability of success for the project should be justified. This can be portrayed as a loop or a spiral where the system to be implemented is considered in more and more detail in each sweep and an evaluation process is undertaken before the next iteration is embarked upon. Figure 4.4 illustrates how SSADM can be interpreted in such a way.

4.10 Software prototyping

A prototype is a working model of one or more aspects of the projected system. It is constructed and tested quickly and inexpensively in order to test out assumptions.

Prototypes can be classified as throw-away, evolutionary or incremental.

Throw-away prototypes Here the prototype is used only to test out some ideas and is then discarded when the development of the operational system is commenced. The prototype could be developed using a different software environment (for example, a 4GL as opposed to a 3GL for the final system where machine efficiency is important) or even on a different kind of hardware platform.

Evolutionary prototypes The prototype is developed and modified until it is finally in a state where it can become the operational system. In this case, the standards that are used to develop the software have to be carefully considered.

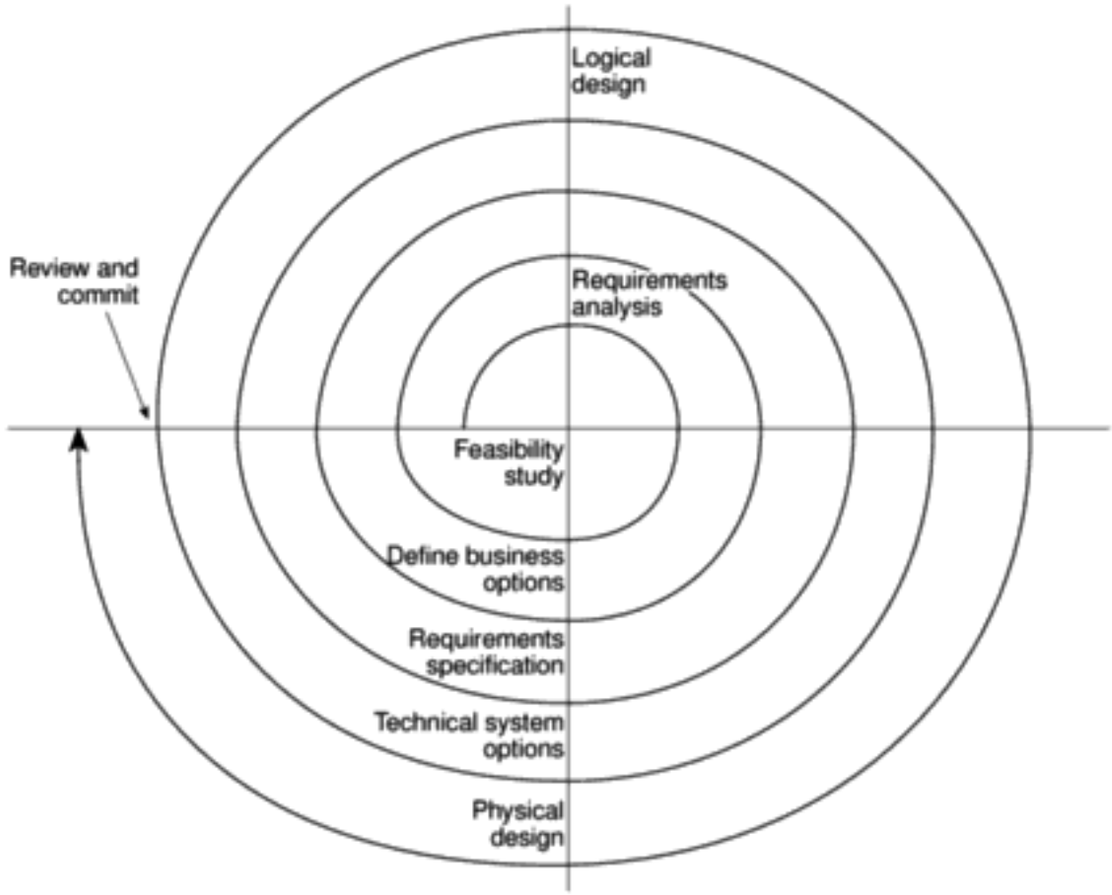


Figure 4.4 *The application of the spiral model to SSADM version 4.*

Incremental prototypes It could be argued that this is, strictly speaking, not prototyping. The operational system is developed and implemented in small stages so that the feed-back from the earlier stages can influence the development of the later stages.

A good book on the general topic of prototyping is R. Vonk's *Prototyping: the effective use of CASE Technology*, Prentice Hall, 1990.

Some of the reasons that have been put forward for prototyping are the following.

Learning by doing When we have just done something for the first time we can usually look back and see where we have made mistakes.

Improved communication Users are often reluctant to read the massive documents produced by structured methods. Even if they do read this documentation, they do not get a feel for how the system is likely to work in practice.

The most important justification for a prototype is the need to reduce uncertainty by conducting an experiment.

Improved user involvement The users may be more actively involved in design decisions about the new system.

Clarification of partially-known requirements Where there is no existing system to mimic, users can often get a better idea of what might be useful to them in a potential system by trying out prototypes.

Demonstration of the consistency and completeness of a specification Any mechanism that attempts to implement a specification on a computer is likely to uncover ambiguities and omissions.

Reduced need for documentation Because a working prototype can be examined, there is less need for detailed documentation. Some may argue, however, that this is a very dangerous suggestion.

Reduced maintenance costs (that is, changes after the system goes live) If the user is unable to suggest modifications at the prototyping stage the chances are that they will ask for the changes as modifications to the operational system. This reduction of maintenance costs is the main plank in the financial case for creating prototypes.

Feature constraint If an application building tool is used, then the prototype will tend to have features that are easily implemented by that tool. A paper-based design might suggest features that are expensive to implement.

Production of expected results The problem with creating test runs is generally not the creation of the test data but the accurate calculation of the expected results. A prototype can be of assistance here.

Software prototyping is not without its drawbacks and dangers, however.

Users sometimes misunderstand the role of the prototype For example, they might expect the prototype to be as robust as an operational system when incorrect data is entered or they might expect the prototype to have as fast a response as the operational system, although this was not the intention.

Lack of project standards possible Evolutionary prototyping could just be an excuse for a sloppy 'hack it out and see what happens' approach.

Lack of control It is sometimes difficult to control the prototyping cycle as the driving force could be the users' propensity to try out new things.

Additional expense Building and exercising a prototype will incur additional expenses. The additional expense might be less than expected, however, because many analysis and design tasks would have to be undertaken anyway. Some research suggests that typically there is a 10% extra cost to produce a prototype.

Machine efficiency A system built through prototyping, while sensitive to the users' needs, might not be as efficient in machine terms as one developed using more conventional methods.

Close proximity of developers Prototyping often means that code developers have to be sited close to the users. One trend has been for organizations in developed countries to get program coding done cheaply in Third World countries such as India. Prototyping generally will not allow this.

4.11 Other ways of categorizing prototypes

What is being learnt?

The most important reason for having a prototype is that there is a need to learn about an area of uncertainty. For any prototype it is essential that the project managers define at the outset what it is intended to learn from the prototype.

This has a particular relevance to student projects. Students often realize that the software that they are to write as part of their project could not safely be used by real users. They therefore call the software a 'prototype'. However, if it is a real prototype then they must:

- specify what they hope to learn from the prototype;
- plan how the prototype is to be evaluated;
- report on what has actually been learnt.

Prototypes may be used to find out how a new development technique can be used. This would be the case where a new methodology is being used in a pilot scheme. Alternatively, the development methods might be well-known, but the nature of the application might be uncertain.

Different projects will have uncertainties at different stages. Prototypes can therefore be used at different stages. A prototype might be used, for instance, at the requirements gathering stage to pin down requirements that seem blurred and shifting. A prototype might, on the other hand, be used at the design stage to test out the users' ability to navigate through a sequence of input screens.

To what extent is the prototyping to be done?

It would be unusual for the whole of the application to be prototyped. The prototyping might take one of the following forms, which simulates only some aspects of the target application.

Mock-ups For example, copies of the screens that the system is to use are shown to the users on a terminal, but the screens cannot actually be used.

Simulated interaction For example, the user can type in a request to access a record and the system will respond by showing the details of a record, but the details shown are always the same and no access is made to a database.

Partial working model:

- *vertical* – some features are prototyped fully
- *horizontal* – all features are prototyped but not in detail (for example, there might not be a full validation of input).

What is being prototyped?

The human–computer interface With information systems, what the system is to do has usually been established and agreed by management at a fairly early

stage in the development of the system. Prototyping tends to be confined to establishing the precise way in which the operators are to interact with the system. In this case, it is important that the physical vehicle for the prototype be as similar as possible to the operational system.

The functionality of the system In other cases, the precise way that the system should function internally will not be known. This will particularly be the case where a computer model of some real-world phenomenon is being developed. The algorithms used need to be repeatedly adjusted until they satisfactorily imitate the behaviour they should be modelling.

Exercise 4.5

At what stage of a system development project (for example, feasibility study, requirements analysis etc.) would a prototype be useful as a means of reducing the following uncertainties?

(a) There is a proposal that the senior managers of an insurance company have personal access to management information through an executive information system installed on personal computers located on their desks. Such a system would be costly to set up and there is some doubt about whether the managers would actually use the system.

(b) A computer system is to support sales office staff who take phone calls from members of the public enquiring about motor insurance and give quotations over the phone.

(c) The insurance company is considering implementing the telephone sales system using the system development features supplied by Microsoft Access. They are not sure, at the moment, that it can provide the kind of interface that would be needed and are also concerned about the possible response times of a system developed using Microsoft Access.

4.12 Tools

Special tools are not essential for prototyping but some have made it more practicable.

Application building tools have many features that allow simple computer-based information systems to be set up quickly so that they can be demonstrated to the staff who will use them. An application written in a 3GL becomes more and more complicated as changes accumulate in its code and this makes the software more difficult to alter safely, while applications implemented using application builders that are often form- and table-driven remain flexible.

It has been suggested that the ease of use of some 4GL application builders and the increasing IT awareness of end users might allow end users to create their own prototypes.

Chapter 9 explores configuration management further.

Where a prototype is being constantly tested by the users and modified by the developers there is a need to be able to control the different versions being produced and where necessary to back-track to previous versions.

4.13 A prototyping example

Details of the study can be found in Mayhew, P. J., Worseley, C. J. & Dearnley, P. A. 'Control of Software Prototyping Process: Change Classification Approach'. *Information and Systems Technology* 13(2) 59-66 published in 1989.

The use of prototyping on the COMET Commercial Estimating System project at the Royal Dockyards was the subject of a study, information about which has been published. The Royal Dockyards had been transferred to a system of 'commercial management' where they had to produce estimates of the cost of doing tasks that could then be compared against the cost of contracting the work out. Staff at the dockyards were inexperienced with this method of working.

It was decided to implement a computer system to support the estimating process and it was realized that the human–computer interface would be important. For this reason, the software house implementing the system suggested a prototyping approach

Among the questions that had to be decided was who should be present at evaluation sessions. If managers were there, would they dominate the proceedings at the expense of those who would actually use the system? Furthermore, if the designers were present they might well feel defensive about the system presented and try and argue against changes suggested.

The results of the prototype

The impact of prototyping may be judged by the fact that although the preliminary design had been done using SSADM, as a result of the evaluation of the prototype the number of screens in the system was doubled.

A major problem was that of controlling changes to the prototype. In order to record and control the changes suggested by users, the changes were categorized into three types.

Cosmetic (about 35% of changes). These were simply changes to the layout of the screen. They were:

- implemented;
- recorded.

Local (about 60% of changes). These involved changes to the way that the screen was processed but did not affect other parts of the system. They were:

- implemented;
- recorded;
- backed-up so that they could removed at a later stage if necessary;
- inspected retrospectively.

Inspections are discussed in Chapter 12.

Global (about 5% of changes), These were changes that affected more than one part of the processing. All changes here had to be the subject of a design review before they could be implemented.

4.14 Incremental delivery

This is similar to the 'incremental prototyping' approach mentioned above. One of the most prominent advocates of this approach is Tom Gilb. The approach involves breaking the system down into small components which are then implemented and delivered in sequence. Each component that is delivered must actually give some benefit to the user. Figure 4.5 gives a general idea of the approach.

Principles of Software Engineering Management by Tom Gilb, published by Addison-Wesley in 1988, argues strongly in favour of this approach.

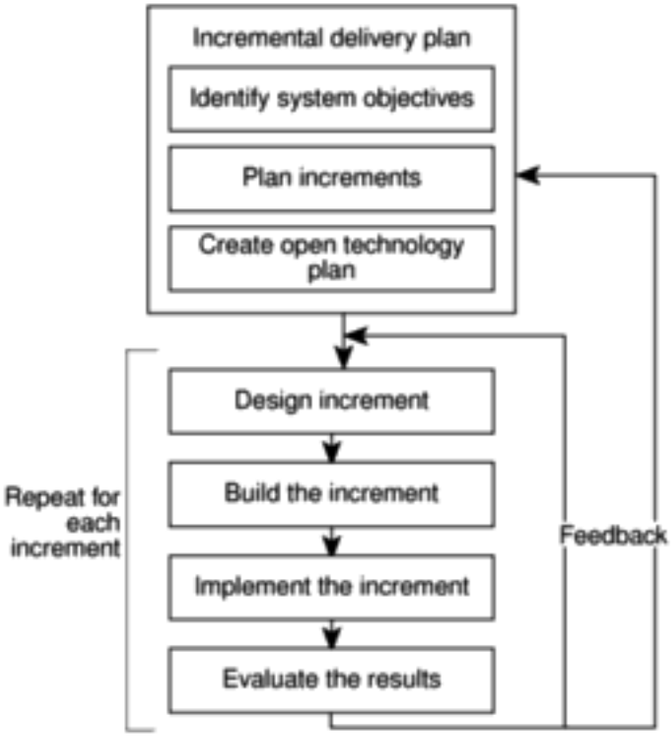


Figure 4.5 *Intentional incremental delivery.*

Advantages of this approach

These are some of the justifications given for the approach:

- the feedback from early increments can influence the later stages;
- the possibility of changes in requirements is not so great as with large monolithic projects because of the shorter timespan between the design of a component and its delivery;
- users get benefits earlier than with a conventional approach;
- early delivery of some useful components improves cash flow, because you get some return on investment early on;

- smaller sub-projects are easier to control and manage;
- 'gold-plating', the requesting of features that are unnecessary and not in fact used, should be less as users will know that they get more than one opportunity to make their requirements known: if a feature is not in the current increment then it can be included in the next;
- the project can be temporarily abandoned if more urgent work crops up;
- job satisfaction is increased for developers who see their labours bearing fruit at regular, short, intervals.

Disadvantages

On the other hand these disadvantages have been put forward:

- 'software breakage', that is, later increments might require the earlier increments to be modified;
- developers might be more productive working on one large system than on a series of smaller ones.

The incremental delivery plan

The process of planning the increments of a project as described by Gilb has similarities with strategic planning described in the previous chapter.

The content of each increment and the order in which the increments are to be delivered to the users of the system have to be planned at the outset.

Basically the same process has to be undertaken as in strategic planning but at a more detailed level where the attention is given to increments of a user application rather than whole applications. The elements of the incremental plan are the *system objectives*, *incremental plan* and the *open technology plan*.

Identify system objectives

The purpose is to give an idea of the 'big picture', that is, the overall objectives that the system is to achieve. These can then expanded into more specific functional goals and quality goals.

Functional goals will include:

- objectives it is intended to achieve;
- jobs the system is to do;
- computer/non-computer functions to achieve them.

Chapter 12 discusses software quality characteristics.

In addition, measurable quality characteristics should be defined, such as reliability, response and security. This reflects Tom Gilb's concern that system developers always keep sight of the objectives that they are trying to achieve on behalf of their clients. In the quickly changing environment of an application, individual requirements may change over the course of the project, but the objectives should not.

Plan increments

Having defined the overall objectives, the next stage is to plan the increments using the following guidelines:

- steps typically should consist of 1% to 5% of the total project;
- non-computer steps may be included – but these must deliver benefits directly to the users;
- ideally, an increment should take one month or less and should never take more than three months;
- each increment should deliver some benefit to the user;
- some increments will be physically dependent on others;
- value-to-cost ratios may be used to decide priorities (see below).

Very often a new system will be replacing an old computer system and the first increments may use parts of the old system. For example, the data for the database of the new system may initially be obtained from the old system's standing files.

Which steps should be first? Some steps might be prerequisites because of physical dependencies but others can be in any order. Value-to-cost ratios can be used to establish the order in which increments are to be developed. The customer is asked to rate each increment with a score in the range 1–10 in terms of its value. The developers also rate the cost of developing each of the increments with a score in the range 0–10. This might seem a rather crude way of evaluating costs and benefits, but people are often unwilling to be more precise. By then dividing the value rating by the cost rating, a rating which indicates the relative 'value for money' of each increment is derived.

The value to cost ratio = V/C where V is a score 1–10 representing value to customer and C is a score 0–10 representing cost.

Table 4.1 *Ranking by value to cost ratio*

Step	*Value*	*Cost*	*Ratio*	*Rank*
Profit reports	9	1	9	(2nd)
Online database	1	9	0.11	(6th)
Ad hoc enquiry	5	5	1	(4th)
Production sequence plans	2	8	0.25	(5th)
Purchasing profit factors	9	4	2.25	(3rd)
Clerical procedures	0	7	0	(7th)
Profit based pay for managers	9	0	∞	(1st)

Create open technology plan

If the system is to be able to cope with new components being continually added then it has to be built so that it is extendible, portable and maintainable.

As a minimum this will require the use of:

- a standard high level language;
- a standard operating system;
- small modules;
- variable parameters, for example, items such as organization name, department names and charge rates are held in a parameter file that can be amended without programmer intervention;
- a standard database management system.

These are all things that might be expected as a matter of course in a modern IS development environment.

4.15 An incremental example

Tom Gilb describes a project where a software house negotiated a fixed price contract with a three-month delivery time with the Swedish Government to supply a system to support map-making. It later became apparent that the original estimate of effort upon which the bid was based was probably about half what the real effort would be.

The project was replanned so that it was divided into ten increments, each supplying something of use to the customer. The final increments were not available until three months after the contract's delivery date. The customer was not in fact unhappy about this as the most important parts of the system had actually been delivered early.

4.16 Selecting the most appropriate process model

See Appendixes C and D for overviews of Euromethod and ISO 12207, both of which offer advice on process model selection.

Both Euromethod and ISO 12207 provide advice on this.

Euromethod distinguishes between the *construction* of an application and its *installation*. It is possible to use different approaches for these two stages. For example, an application could be constructed using a one-shot strategy but then be released to its users in increments. The only combinations of construction and installation strategies that are not feasible are evolutionary installation with any other construction approach than evolutionary.

In general, Euromethod suggests that where uncertainty is high then an evolutionary approach is to be favoured. An example of uncertainty would be where the users' requirements are not clearly defined. Where the requirements are relatively certain but there are many complexities, for example, where there is a large embedded system that is going to need a large amount of code, then an incremental approach might be favoured. Where deadlines are tight, then either an evolutionary or an incremental approach is favoured over a one-shot strategy, as both tactics should allow at least something to be delivered at the deadline, even

if it is not all that was originally promised. Students about to plan final year projects would do well to note this.

4.17 Conclusion

This chapter has stressed the need to examine each project carefully to see if it has characteristics that suggest a particular approach or process model. These characteristics will also suggest the addition of specific activities to the project plan.

The classic waterfall process model, which attempts to minimize iteration, should lead to projects that are easy to control. Unfortunately many projects do not lend themselves to this structure. Prototyping often reduces project uncertainties by allowing knowledge to be bought through experimentation. The incremental approach encourages the execution of a series of small, manageable, 'mini-projects' but does have some costs.

4.18 Further exercises

1. A bank which provides loans to home buyers has a long history of implementing computer-based information systems to support the work of its branches. It uses a proprietary structured systems analysis and design method. It has been decided to create a computer model of the property market. This would attempt for example to calculate the effect of changes of interest rates on house values. There is some concern that the usual methodology used for IS development would not be appropriate for the new project.

   (a) Why might there be this concern and what other approaches should be considered?
   (b) Outline a plan for the development of the system that illustrates the application of your preferred methodology for this project.

2. A software application is to be designed and built to assist in software cost estimation. It responds to certain input parameters and produces initial cost estimates to be used at bidding time.

   (a) It has been suggested that a software prototype would be of value in these circumstances. Explain why this might be.
   (b) Discuss how such prototyping could be controlled to ensure that it is conducted in an orderly and effective way and within a specified time span.

3. An invoicing system is to have the following components: amend invoice, produce invoice, produce monthly statements, record cash payment, clear paid invoices from database, create customer records, delete customer.

   (a) What physical dependencies govern the order in which these transactions are implemented?

(b) How could the system be broken down into increments which would be of some value to the users. Hint – think about the problems of taking existing details onto a database when a system is first implemented.

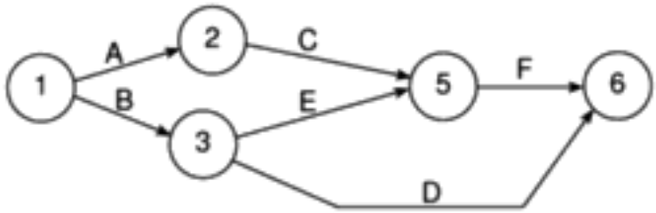
4. What are the features of the following that contribute to an open systems architecture as recommended by Tom Gilb:

   (a) the UNIX™ operating system;
   (b) SQL;
   (c) C++;
   (d) Jackson Structured Programming.

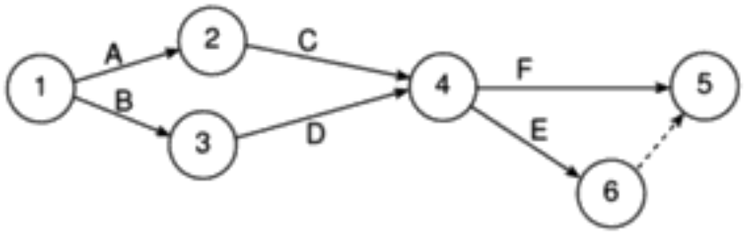
Chapter 6

6.1 Errors drawing activity networks

(a) Activity D dangles, giving the project two 'end events'. This network should be drawn as below. To aid comparison with the original, the nodes have not been renumbered, although we would normally do so.

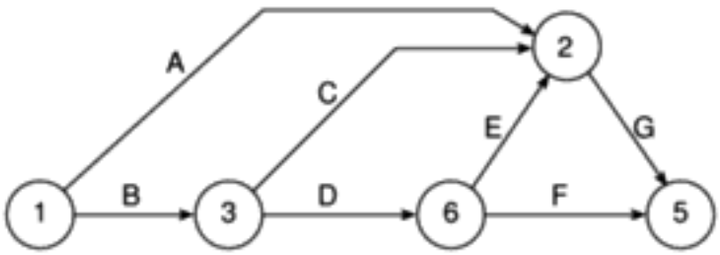


(b) Once again, this network has two end nodes, but in this case the solution is slightly different since we should introduce a dummy activity if we are to follow the standard CPM conventions.



(c) Either this one has a dangle (although, because of the way it is drawn, it is less obvious) or activity E has its arrow pointing in the wrong direction. We need a bit more information before we can redraw this one correctly.

(d) Strictly speaking, there is nothing wrong with this one – it is just badly drawn and the nodes are not numbered according to the standard conventions. It should be redrawn as in the following example.



In this diagram the nodes have retained their original numbers (to aid identification) although they should of course be renumbered sequentially from left to right.

(e) This one contains a loop – F cannot start before G has finished, G cannot start before E has finished and E cannot start before G has finished. One of the arrows is wrong! It is probably activity F that is wrong but we cannot be sure without further information.

Global Journal of Computer Science & Technology
Volume 11 Issue 5 Version 1.0 April 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
ISSN: 0975-5861

The Role and Impact of Project Management in ERP project implementation life cycle

By Anees Ara, Abdullah S. Al-Mudimigh
King Saud University

Abstract- : Recent advancement of Information Technology in business management processes has flourished ERP as one of the most widely implemented business software systems in variety of industries and organizations. This paper presents review on the impact of project management in ERP project life cycle by studying various project management methodologies. Also the role and critical activities of project manager, project team and hence project management is explored in ERP projects implementation in organization of different sizes and culture.

Keywords: ERP, Project Management, Implementation phase, organizational culture, Risk.

Classification: *GJCST Classification: J.1, K.3.m*

Strictly as per the compliance and regulations of:

The Role and Impact of Project Management in ERP project implementation life cycle

Anees Ara[α], Abdullah S. Al-Mudimigh[Ω]

Abstract- Recent advancement of Information Technology in business management processes has flourished ERP as one of the most widely implemented business software systems in variety of industries and organizations. This paper presents review on the impact of project management in ERP project life cycle by studying various project management methodologies. Also the role and critical activities of project manager, project team and hence project management is explored in ERP projects implementation in organization of different sizes and culture.

KeyWords: ERP, Project Management, Implementation phase, organizational culture, Risk.

I. Introduction

ERP is a system for the seamless integration of all the information flowing through the company such as finances, accounting, human resources, supply chain, and customer information [8]. The selection, procurement, and deployment of an ERP system are nothing but involvement of high risks in exchange for significant business and financial rewards [12]. A successful ERP system can be the backbone of business intelligence for an organization because it can give managers an integrated view of the processes involved within it [14]. The effective ERP implementation brings in reduction of cost improvement in quality, productivity & customer service, better resource management, improved decision-making, planning and hence organizational empowerment [15]. Despite the wide spread applications and benefits of ERP systems, the statistics show that about 30% of ERP implementations have been successful [16]. In order to overcome failures through comprehensive literature review 11 critical factors for successful implementation of enterprise systems were identified as ERP teamwork and composition; change management program and culture; top management support; business plan and vision; business process reengineering with minimum customization; project management; monitoring and evaluation of performance; effective communication; software development, testing and troubleshooting; project champion; appropriate business and IT legacy sysFactstems .

About [α]- Department of Information Technology College of Computer and Information Sciences King Saud University
Email: aahamed@ksu.edu.sa
About [Ω]- Department of Information Systems College of Computer and Information Sciences King Saud University
Email: mudimigh@ksu.edu.sa

II. Project Management Methodologies and Techniques

ERP is a management mode or techniques. Many companies regard ERP system implementation as a project management [7]. The implementation of ERP projects involves various management functions, which inevitably leads to different levels of management reorganizations [9]. Successful project management is about managing the risk. Project Management is the application of knowledge, skills, tools, and techniques to project activities to meet project requirements. This can be done by using a formal project management structure (PMS). One of the worlds famous PMS is project management body of knowledge (PMBOK), which is developed by Project management institute (PMI). This methodology includes 5 processes/phases of project management such as project initiation, planning, execution, control, and closing. Also it is comprised by 9 knowledge areas such as Project Integration Management, Project Scope Management, Project Time Management, Project Cost Management, Project Quality Management, Project Human Resources Management, Project Communications Management, Project Risk Management, and Project Procurement Management. The results from the study on project management strategies under PMBOK frame work by Fergal et- al[3], illustrates PMBOK as bit fit for ERP projects and also reveals that the importance of project governance and the need for a multi-level structure spanning both the cooperate and local levels. These structures would ensure the project to be directed properly & focused, and to reduce delays & rework due to the fact that timely problem resolution could be carried out [3].

Since ERP environment faces constant change and reassessment of organizational processes and technology [18] and hence the project management method used with ERP deployments must provide adaptability and agility to support these evolutionary process and technology [17]. The major problem with software development/deployment is managerial, but not technical. Therefore a method of augmenting structured project methods with agility to produce a new approach to managing projects is proposed by Alleman [12]. This method is based on venture capitalist approach which includes: staged investments, managed risks & people (team involved). The author discovered that traditional IT project management waterfall model which includes

planning, change and stability had several erroneous assumptions. However he suggested that through planning in the presence of uncertainty, avoiding dysfunctional relationships and improper pretentions, we can overcome the flaws in traditional approach. By applying agile values, author proposed the following principles for managing ERP projects in agile manner: assume simplicity, embrace change, enabling the next effort, incremental change, maximize stakeholder value, manage with a purpose, multiple views, rapid feedback, working software is the primary goal and travel light[12].

Email: mudimigh@ksu.edu.sa

III. Role of Project Manager

Jones et-al [19] explained that the first six out of sixteen technology factors associated with software disasters are specific failures in the domains of project management, and three of the other technology deficiencies can be indirectly assigned to poor management practices. The top 10 ERP management headaches rank issues are project size, staffing, risk management, unreasonable deadlines, funding, organizational politics, scope creep, unexpected gaps, interfaces, and resistance to change [20]. One of the basic principles for assuring the ERP implementation success is the appointment of the project manager, who should be most talented business manager. The characteristics of a successful project manager are flexible, disciplined, quick learner, good decision maker, ERP expert, having business experience, political clout, good formal education, well liked, and motivates staff [23]. In order to make ERP project successful, the project manager must work efficiently in the following areas: deciding on project scope, managing risks, discovering gaps, the right staff, preventing brain drain, project scheduling, interface with other systems, monitoring progress, and managing chaos[20]. In addition, the focused management by project manager of the following critical activities that streamline the ERP project management will enhance the success of any ERP project: Embrace overall goals and objectives, defining requirements, review as is – to be, use proto education, business system test/conference room pilot(CRP), execute timely cut-over/conversion processes, and going live and beyond[11].

IV. The Impact of Project Management in ERP Implementation

The strategic project management in [system evolution] enables success by first detecting the need for change in all relevant corners of the company, and second, servicing the need with appropriate resources. By actively addressing the change needs of people, process, and technology, the SPM will ensure the system that is deployed meets tenets of overall business strategy while also becoming the enabler of success.

a) *Pre implementation*

In the article [21], the author proposes that the key to successful ERP implementation is through the use of project management life cycle theory analysis of various stages of ERP implementation, and it cannot be considered lead to failure. He supports his stand by introducing the Tasly ERP projects in the pre project research, project organization and project management in the process of successful experience. Throughout the project management concepts and methods of operation, ultimately leads to success of ERP system and enhances the overall enterprise management level. And due to comprehensive survey & evaluation, efficient project management team, sound project management information and because of the need and the change continues to improve the Tasly was successful in implementing ERP.

b) *During implementation*

Tsai et al[7], through an immense literature review identified and categorized 8 achievement level of project management as (1) fulfilling business implementation goal (2) full of top management support (3) meeting schedule goal (4) meeting budget objective (5) triggering effective communication (6) solving problem (7) fulfilling integration of system and (8) user acceptance[7]. The study and the empirical investigation on consultant criteria, project management and performance enhancement indicated that these three factors are integrated and the service quality one affects the other and hence adds to performance enhancement of the ERP implementation [7].

c) *Post implementation*

Ying shi [12], through a case study in application research of project management in ERP system implementation process, revealed that the Project management theory and methods were used in the construction of enterprise information, in line with the overall planning, step by step principles for the business to the ERP project involves all aspects of effective planning, organization, management and monitoring, thus to achieve the desired goals and effect for the enterprise benefits.

d) *Organizational culture*

Public and private sectors: The recent years have shown a tremendous growth in application of ERP systems in public and government sectors. It is observed that public leaders are more concerned in implementing the best practices for each business process provided by ERP. In general the ERP project implementation is based on the three factors as people, process and technology. But the IT based challenges in government sectors are mostly human than technical. And one of the main causes among them is poor project management [5]. From the study on Jordanian culture in [4] the author described that there is a significant difference in project

management aspects, in ERP implementation, when applied to public and private sectors. Moreover, since the critical decisions and approvals are taken only by top management and due to rigid hierarchy and structures, more bureaucracy and delay in decision making in public organizations leads to affect timely implementation [4].

Large and SMEs: Through the appropriate use of Characteristic Analysis Method (CAM), a tool to ensure that the IT project is manageable and consistent by its different goals content and development approaches, a case study on three SMEs, the author shown inadequacies in the fields of management and leadership that the implementation of ERP system causes risks in companies [15].

e) *Risk Management*

ERP projects implementations are complex, resource-intensive and risky. And risks can be mitigated through strong executive sponsorship, communication and involvement of stakeholders, and good project management [5]. Davide et- al[6] classified ERP project failure into four levels as process failure, expectation failure, interaction failure, and correspondence failure, which directly relates to the critical activities of project management team as explained earlier in this paper [6]. Hence an emphasis must be placed on project team selection and project manager's pursuance should be on specific areas of ERP life cycle, i.e. if any unclear area not solved in exploration cycle of ERP project must be taken care in implementation else this would remerge post go-live with disastrous consequences [3].

V. Conclusion

By studying various project management methodologies and techniques, and also through various literatures, it is found that in ERP project life cycle, the project management plays a key role and hence a proper emphasis must be placed in selecting the project team that ensures proper decision making and results in timely project completion. Hence by applying the theory and method of project management life cycle on project implementation will consequently endure success .The future research can be carried out in enhancing ERP implementation through web 2.0 technologies and also Lean management, which indeed suffice with ERP software.

References Références Referencias

1. Nah. F.F., Lau. L. J., Kuang. J., (2001), Critical factors for successful implementation of enterprise systems", Business Process Management Journal, Vol. 7 Iss: 3, pp.285 – 296
2. Reimers. K., (2002), "Implementing ERP Systems in China", in Proceedings of the 35 th Hawaii International Conference on Systems Sciences, Hawaii Rockart, JF (1979), 'Chief Executives Define Their Own Needs', Harvard Business Review
3. Carton. F., Adam F., Sammon D., (2008), "Project management: a case study of a successful ERP implementation", International Journal of Managing Projects in Business, Vol. 1 Iss: 1, pp.106 – 124.
4. Rabaa'i. Ahmad A. R., (2009), "The impact of organisational culture on ERP systems implementation: lessons from Jordan". In: Proceedings of the Pacific Asia Conference on Information Systems 2009, 10-12 July 2009, Hyderabad, India.
5. "Implementing ERP Systems in the Public Sector: Nine Sure Ways to Failùor Succeed" is a White paper, Strategic Management Tools BearingPoint, CBS interactive 2010. http://jobfunctions.bnet.com/abstract.aspx?docid=143127
6. Aloini. D., Dulmin. R., Mininno. V., (2007) "Risk management in ERP project introduction: Review of the literature", Information & Management Vol. 44, No. 6. pp. 547-567.
7. Tsai. W.H., Shen. Y. S., Lee. P. L., Kuo. L., (2009). "An empirical investigation of the impacts of ERP consultant selections and project management on ERP is success assessment", Industrial Engineering and Engineering Management, 2009, IEEM. IEEE International Conference on, pp: 568-572, Hong Kong.
8. Davenport. T., (1998), "Putting the enterprise into the enterprise system", Harvard Business Review, July-August, pp.121- 131.
9. Shi. Y., (2010), "Application research of project management in ERP system implementation process", Emergency Management and Management Sciences (ICEMMS), 2010 IEEE International Conference on, pp: 68 - 71, Beijing.
10. PMI (2000), A Guide to the Project Management Body of Knowledge, PMI Publishing Division, The Project Management Institute, Sylva, NC.
11. Schwartz. T., (2005), "ERP Project Management An activity based approach", CPIM,IT toolbox ERP. http://hosteddocs.ittoolbox.com/TS061705.pdf.
12. Alleman. G.B., (2002), "Agile Project Management Methods for ERP: How to Apply Agile Processes to Complex COTS Projects and Live to Tell About It". In XP/Agile Universe LNCS 2418.
13. Beheshti. H., (2006), "What managers should know about ERP/ERP II", Management Research News, 29, (4), 184-193.

14. Parr. A., Shanks. G., (2000), "A model of ERP project implementation", Journal of Information Technology, 15 (4), 289–303
15. Iskanius. P., (2004) "Risk management in ERP project in the context of SME", Engineering Letters
16. Standish Group International, Inc., the Quarter Research Report, 2004
17. Earl, Michael, Sampler. J., Short. J., (1995), "Strategies for Reengineering: Different ways of Initiating and Implementing Business Process Change," Centre for Research in Information Management, London Business School.
18. Ross, Jeanne (1999), "Surprising Facts About Implementing ERP," IT Pro, pp. 65–68.
19. Jones, Capers. (1996), "Patterns of Software Systems Failures and Success", International Thompson Computer Press.
20. Trepper. C., (1999), "ERP Project Management Is Key to a Successful Implementation", http://itmanagement.earthweb.com/entdev/article.php/614681
21. (2007) "Organizational Project Management - ERP Tasly road to success", enterprise research papers, http://eng.hi138.com/?i142759
22. CFO.com (2009) "The ABC's of ERP", CFO IT, Accenture, SAP, CFO Publishing Corporation.
23. Gartner Institute, Gartner Group, www.gartner.com.

Chapter 9

Monitoring and control

OBJECTIVES

When you have completed this chapter you will be able to:

- monitor the progress of projects;
- assess the risk of slippage;
- visualize and assess the state of a project;
- revise targets to correct or counteract drift;
- control changes to a project's requirements.

9.1 Introduction

Once work schedules have been published and the project is under way, attention must be focused on ensuring progress. This requires monitoring of what is happening, comparison of actual achievement against the schedule and, where necessary, revision of plans and schedules to bring the project as far as possible back on target.

In earlier chapters we have stressed the importance of producing plans that can be monitored – for example, ensuring that activities have clearly defined and visible completion points. We will discuss how information about project progress is gathered and what actions must be taken to ensure a project meets its targets.

The final part of this chapter discusses how we can deal with changes that are imposed from outside – namely, changes in requirements.

9.2 Creating the framework

Exercising control over a project and ensuring that targets are met is a matter of regular monitoring, finding out what is happening, and comparing it with current

targets. If there is a mismatch between the planned outcomes and the actual ones then either replanning is needed to bring the project back on target or the target will have to be revised. Figure 9.1 illustrates a model of the project control cycle and shows how, once the initial project plan has been published, project control is a continual process of monitoring progress against that plan and, where necessary, revising the plan to take account of deviations. It also illustrates the important steps that must be taken after completion of the project so that the experienced gained in any one project can feed into the planning stages of future projects, thus allowing us to learn from past mistakes.

See Chapter 11 for a discussion of software quality.

In practice we are normally concerned with departures from the plan in four dimensions – delays in meeting target dates, shortfalls in quality, inadequate functionality, and costs going over target. In this chapter we are mainly concerned with the first and last of these.

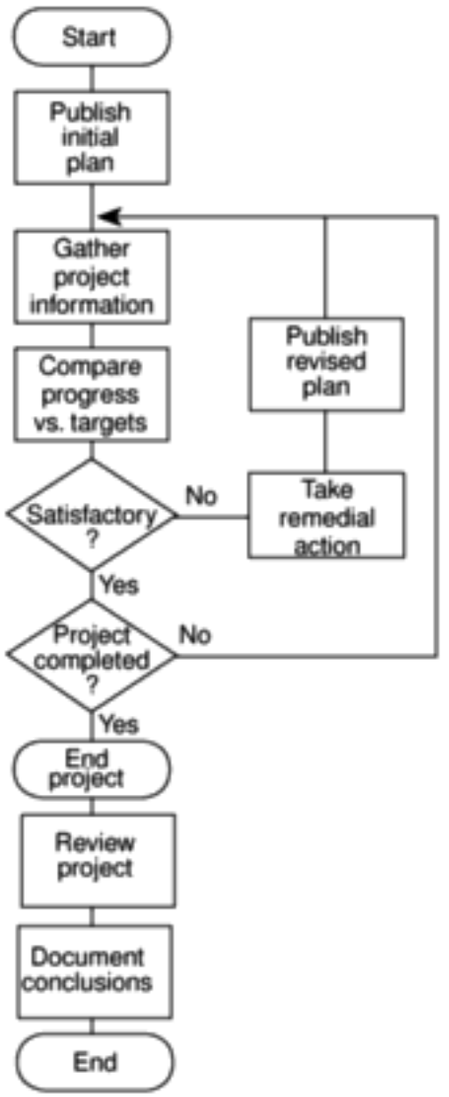


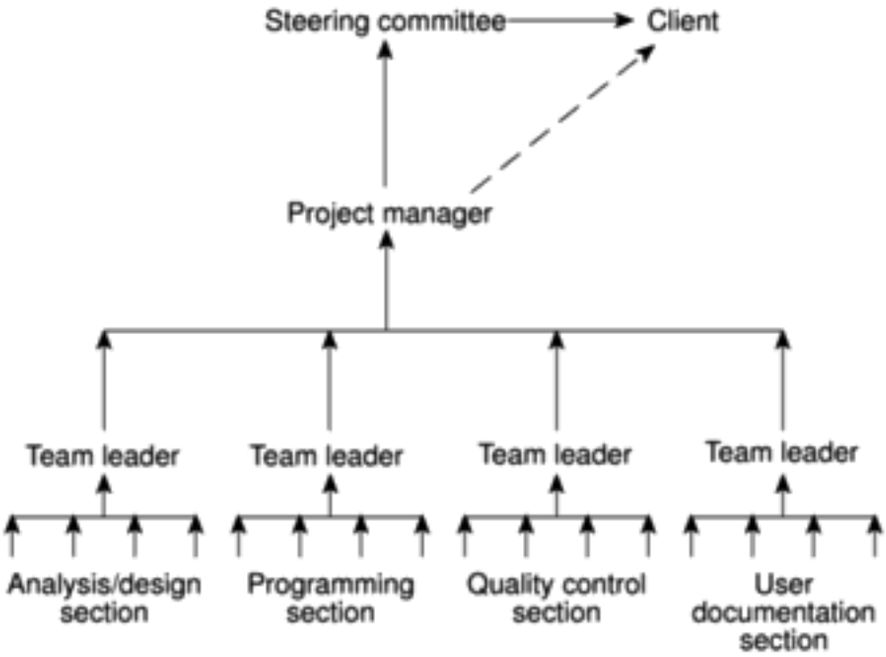
Figure 9.1 *The project control cycle.*

Responsibility

The overall responsibility for ensuring satisfactory progress on a project is often the role of the *project steering committee* or *Project Board*. Day-to-day responsibility will rest with the project manager and, in all but the smallest of projects, aspects of this can be delegated to team leaders.

Figure 9.2 illustrates the typical reporting structure found with medium and large projects. With small projects (employing around half a dozen or fewer staff) individual team members usually report directly to the project manager, but in most cases team leaders will collate reports on their section's progress and forward summaries to the project manager. These, in turn, will be incorporated into project-level reports for the steering committee and, via them or directly, progress reports for the client.

The concept of a reporting hierarchy was introduced in Chapter 1.



In a PRINCE 2 environment, there is a Project Assurance function reporting to the Project Board and independent of the Project Manager.

Figure 9.2 *Project reporting structures.*

Reporting may be oral or written, formal or informal, or regular or ad hoc and some examples of each type are given in Table 9.1. While any effective team leader or project manager will be in touch with team members and available to discuss problems, any such informal reporting of project progress must be complemented by formal reporting procedures – and it is those we are concerned with in this chapter.

Assessing progress

Progress assessment will normally be made on the basis of information collected and collated at regular intervals or when specific events occur. Wherever possible, this information will be objective and tangible – whether or not a particular report has been delivered, for example. However, such end-of-activity deliverables might

Table 9.1 *Categories of reporting*

Report type	*Examples*	*Comment*
Oral formal regular	weekly or monthly progress meetings	while reports may be oral formal written minutes should be kept
Oral formal ad hoc	end-of-stage review meetings	while largely oral, likely to receive and generate written reports
Written formal regular	job sheets, progress reports	normally weekly using forms
Written formal ad hoc	exception reports, change reports	
Oral informal ad hoc	canteen discussion, social interaction	often provides early warning; must be backed up by formal reporting

not occur sufficiently frequently throughout the life of the project. Here progress assessment will have to rely on the judgement of the team members who are carrying out the project activities.

Setting checkpoints

The PRINCE 2 standard described in Appendix A has its own terminology.

It is essential to set a series of checkpoints in the initial activity plan. Checkpoints may be:

- regular (monthly, for example);
- tied to specific events such as the production of a report or other deliverable.

Taking snap-shots

The frequency with which the a manager needs to receive information about progress will depend upon the size and degree of risk of the project or that part of the project under their control. Team leaders, for example, need to assess progress daily (particularly when employing inexperienced staff) whereas project managers may find weekly or monthly reporting appropriate. In general, the higher the level, the less frequent and less detailed the reporting needs to be.

Short, Monday morning team progress meetings are a common way of motivating staff to meet short term targets.

There are, however, strong arguments in favour of formal weekly collection of information from staff carrying out activities. Collecting data at the end of each week ensures that information is provided while memories are still relatively fresh and provides a mechanism for individuals to review and reflect upon their progress during the past few days.

Major, or project-level, progress reviews will generally take place at particular points during the life of a project – commonly known as *review points* or *control points*. PRINCE 2, for example, designates a series of checkpoints where the status of work in a project or for a team is reviewed. At the end of each project Stage, PRINCE 2 provides for an End Stage Assessment where an assessment of the project and consideration of its future are undertaken.

9.3 Collecting the data

As a rule, managers will try to break down long activities into more controllable tasks of one or two weeks duration. However, it will still be necessary to gather information about partially completed activities and, in particular, forecasts of how much work is left to be completed. It can be difficult to make such forecasts accurately.

Exercise 9.1

A software developer working on Amanda's project has written the first 250 lines of a Cobol program that is estimated to require 500 lines of code. Explain why it would be unreasonable to assume that the programming task is 50% complete.

How might you make a reasonable estimate of how near completion it might be?

Where there is a series of products, partial completion of activities is easier to estimate. Counting the number of record specifications or screen layouts produced, for example, can provide a reasonable measure of progress.

In some cases, intermediate products can be used as in-activity milestones. The first successful compilation of a Cobol program, for example, might be considered a milestone even though it is not the final product of the activity code and test.

Partial completion reporting

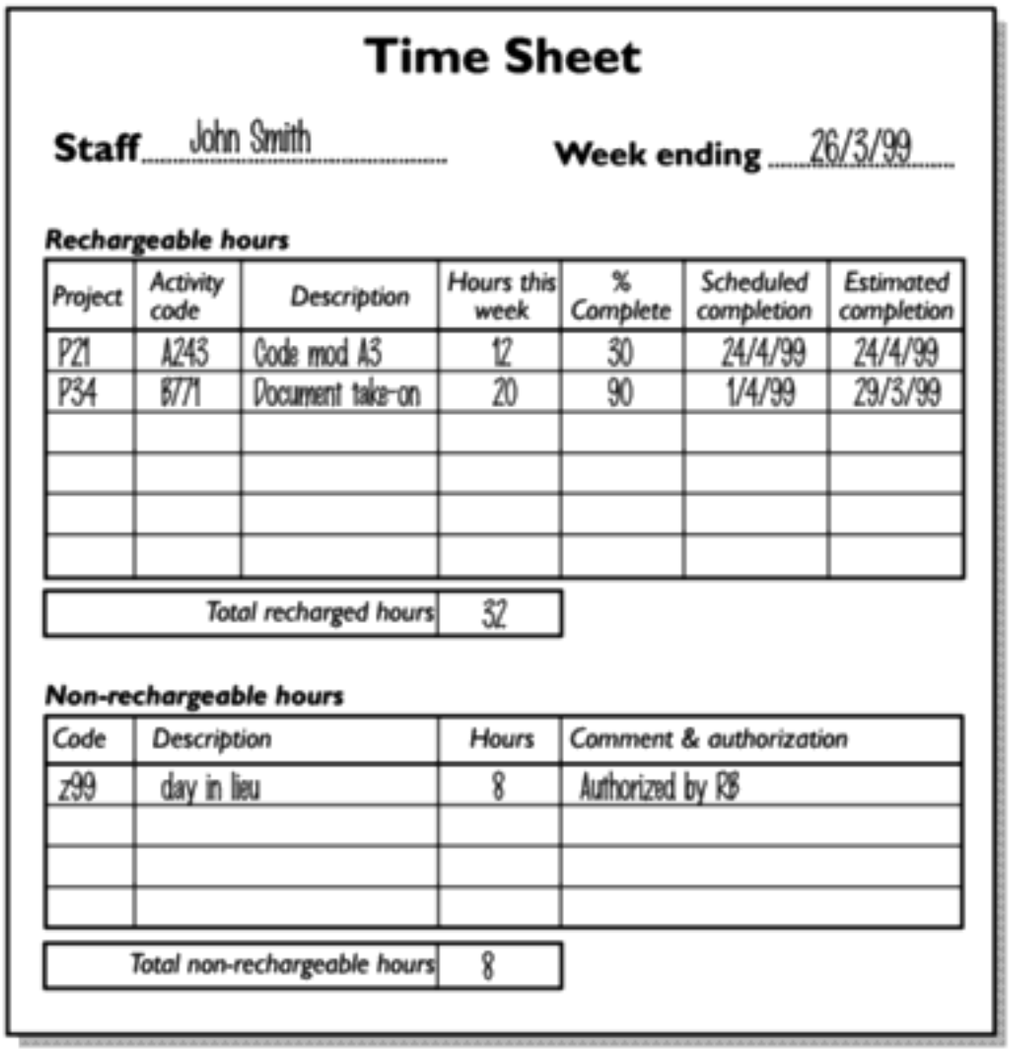
Many organizations use standard accounting systems with weekly time sheets to charge staff time to individual jobs. The staff time booked to a project indicates the work carried out and the charges to the project. It does not, however, tell the project manager what has been produced or whether tasks are on schedule.

It is therefore common to adapt or enhance existing accounting data collection systems to meet the needs of project control. Weekly time sheets, for example, are frequently adapted by breaking jobs down to activity level and requiring information about work done in addition to time spent. Figure 9.3 shows a typical example of such a report form, in this case requesting information about likely slippage of completion dates as well as estimates of completeness.

Asking for estimated completion times can be criticized on the grounds that frequent invitations to reconsider completion dates deflects attention away from the importance of the originally scheduled targets and can generate an ethos that it is acceptable for completion dates to slip.

Weekly timesheets are a valuable source of information about resources used.

They are often used to provide information about what has been achieved. However, requesting partial completion estimates where they cannot be obtained from objective measures encourages the 99% complete syndrome – tasks are reported as on time until 99% complete, and then stay at 99% complete until finished.



Time Sheet

Staff John Smith **Week ending** 26/3/99

Rechargeable hours

Project	Activity code	Description	Hours this week	% Complete	Scheduled completion	Estimated completion
P21	A243	Code mod A3	12	30	24/4/99	24/4/99
P34	B771	Document take-on	20	90	1/4/99	29/3/99

Total recharged hours	32

Non-rechargeable hours

Code	Description	Hours	Comment & authorization
z99	day in lieu	8	Authorized by RB

Total non-rechargeable hours	8

Figure 9.3 *A weekly time sheet and progress review form.*

Risk reporting

One popular way of overcoming the objections to partial completion reporting is to avoid asking for estimated completion dates, but to ask instead for the team members' estimates of the likelihood of meeting the planned target date.

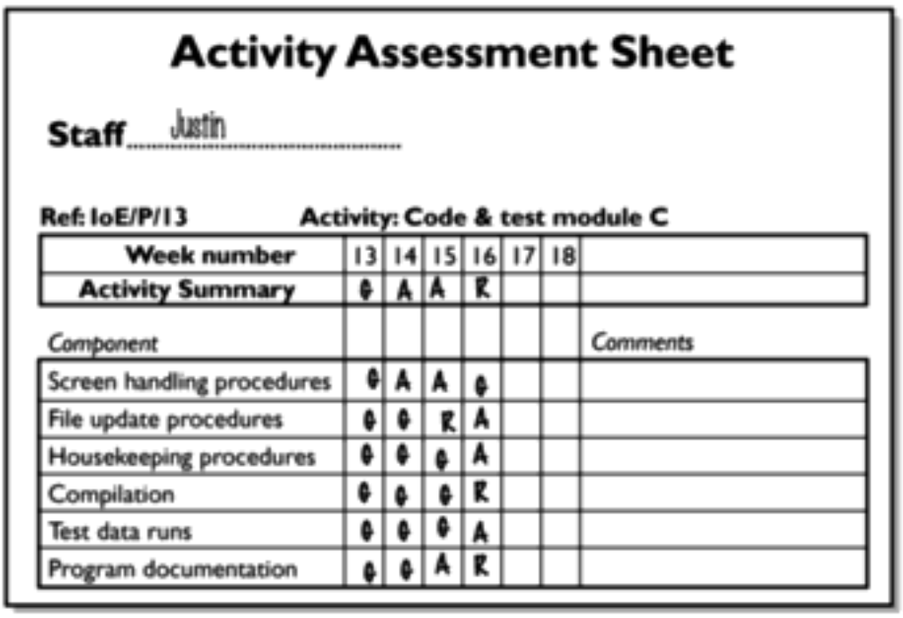
There are a number of variations on the traffic-light technique. The version described here is in use in IBM and is described in Down, Coleman and Absolon, *Risk Management for Software Projects*, McGraw-Hill, 1994.

One way of doing this is the traffic-light method. This consists of the following steps:

- identify the key (first level) elements for assessment in a piece of work;
- break these key elements into constituent elements (second level);
- assess each of the second level elements on the scale *green* for 'on target', *amber* for 'not on target but recoverable', and *red* for 'not on target and recoverable only with difficulty';
- review all the second level assessments to arrive at first level assessments;
- review first and second level assessments to produce an overall assessment.

For example, Amanda decides to use a version of the traffic-light method for reviewing activities on the IOE project. She breaks each activity into a number of component parts (deciding, in this case, that a further breakdown is unnecessary) and gets the team members to complete a return at the end of each week. Figure 9.4 illustrates Justin's completed assessment at the end of week 16.

Note that this form refers only to uncompleted activities. Justin would still need to report activity completions and the time spent on activities.



Activity Assessment Sheet

Staff Justin

Ref: IoE/P/13 **Activity: Code & test module C**

Week number	13	14	15	16	17	18	
Activity Summary	G	A	A	R			
Component							*Comments*
Screen handling procedures	G	A	A	G			
File update procedures	G	G	R	A			
Housekeeping procedures	G	G	G	A			
Compilation	G	G	G	R			
Test data runs	G	G	G	A			
Program documentation	G	G	A	R			

Figure 9.4 *A traffic-light assessment of IoE/P/13.*

Traffic-light assessment highlights only risk of non-achievement; it is not an attempt to estimate work done or to quantify expected delays.

Following completion of assessment forms for all activities, the project manager uses these as a basis for evaluating the overall status of the project. Any critical activity classified as amber or red will require further consideration and often leads to a revision of the project schedule. Non-critical activities are likely to be considered as a problem if they are classified as red, especially if all their float is likely to be consumed.

9.4 Visualizing progress

Having collected data about project progress, a manager needs some way of presenting that data to greatest effect. In this section, we look at some methods of presenting a picture of the project and its future. Some of these methods (such as Gantt charts) provide a static picture, a single snap-shot, whereas others (such as time-line charts) try to show how the project has progressed and changed through time.

The Gantt chart

One of the simplest and oldest techniques for tracking project progress is the Gantt chart. This is essentially an activity bar chart indicating scheduled activity dates

and durations frequently augmented with activity floats. Reported progress is recorded on the chart (normally by shading activity bars) and a 'today cursor' provides an immediate visual indication of which activities are ahead or behind schedule. Figure 9.5 shows part of Amanda's Gantt chart as at the end of Tuesday of week 17. *Code & test module D* has been completed ahead of schedule and *code & test module A* appears also to be ahead of schedule. The coding and testing of the other two modules are behind schedule.

Henry Gantt (1861–1919) was an industrial engineer interested in the efficient organization of work.

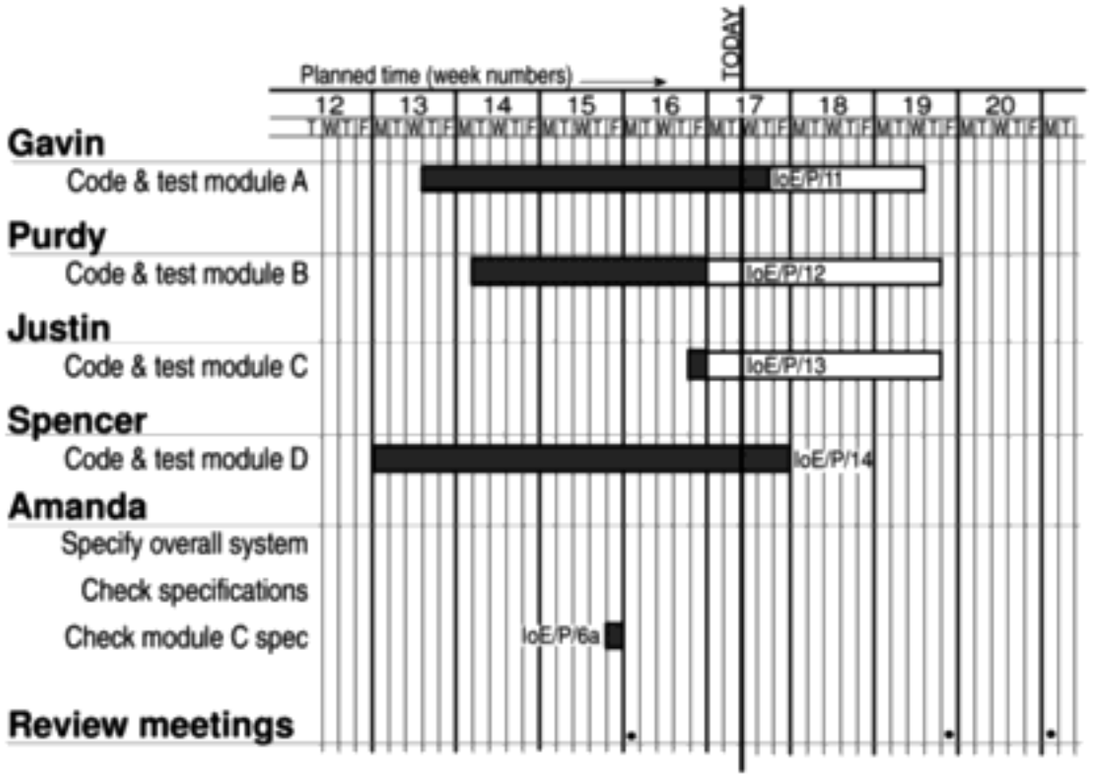


Figure 9.5 *Part of Amanda's Gantt chart with the 'today cursor' in week 17.*

The slip chart

A slip chart (Figure 9.6) is a very similar alternative favoured by some project managers who believe it provides a more striking visual indication of those activities that are not progressing to schedule – the more the slip line bends, the greater the variation from the plan. Additional slip lines are added at intervals and, as they build up, the project manager will gain an idea as to whether the project is improving (subsequent slip lines bend less) or not. A very jagged slip line indicates a need for rescheduling.

Ball charts

A somewhat more striking way of showing whether or not targets have been met is to use a ball chart as in Figure 9.7. In this version of the ball chart, the circles indicate start and completion points for activities. The circles initially contain the original scheduled dates. Whenever revisions are produced these are added as second dates in the appropriate circle until an activity is actually started or

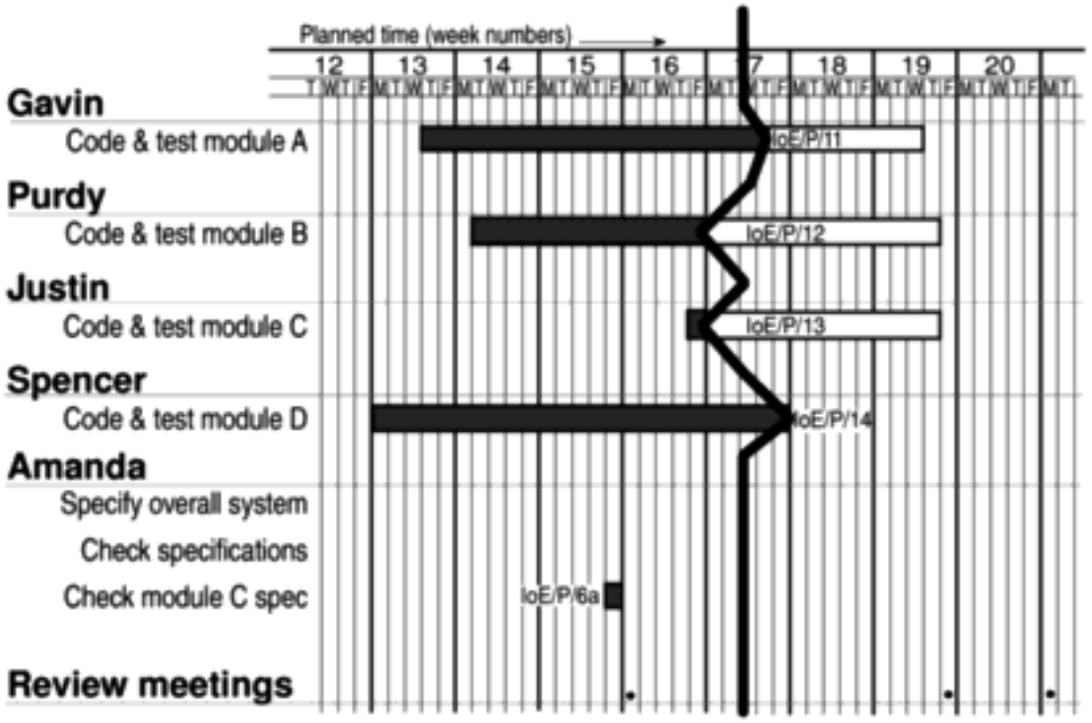


Figure 9.6 *The slip chart emphasizes the relative position of each activity.*

completed when the relevant date replaces the revised estimate (in bold italic in Figure 9.7). Circles will therefore contain only two dates, the original and most recent target dates, or the original and actual dates.

Where the actual start or finish date for an activity is later than the target date, the circle is coloured red (dark grey in Figure 9.7) – where an actual date is on time or earlier than the target then the circle is coloured green (light grey in Figure 9.7).

David Youll in *Making Software Development Visible*, John Wiley & Sons, 1990, describes a version of the ball chart using three sets of dates and part-coloured balls.

Such charts are frequently placed in a prominent position and the colour coded balls provide a constant reminder to the project team. Where more than one team is working in close proximity, such a highly visible record of achievement can encourage competitiveness between teams.

Another advantage of ball charts over Gantt and slip charts is that they are relatively easy to keep up to date – only the dates and possibly colours need to be changed, whereas the others need to be redrawn each time target dates are revised.

The timeline

One disadvantage of the charts described so far is that they do not show clearly the slippage of the project completion date through the life of the project. Knowing the current state of a project helps in revising plans to bring it back on target, but analysing and understanding trends helps to avoid slippage in future projects.

The timeline chart is a method of recording and displaying the way in which targets have changed throughout the duration of the project.

Figure 9.8 shows a timeline chart for Brigette's project at the end of the sixth week. Planned time is plotted along the horizontal axis and elapsed time down the vertical axis. The lines meandering down the chart represent scheduled activity

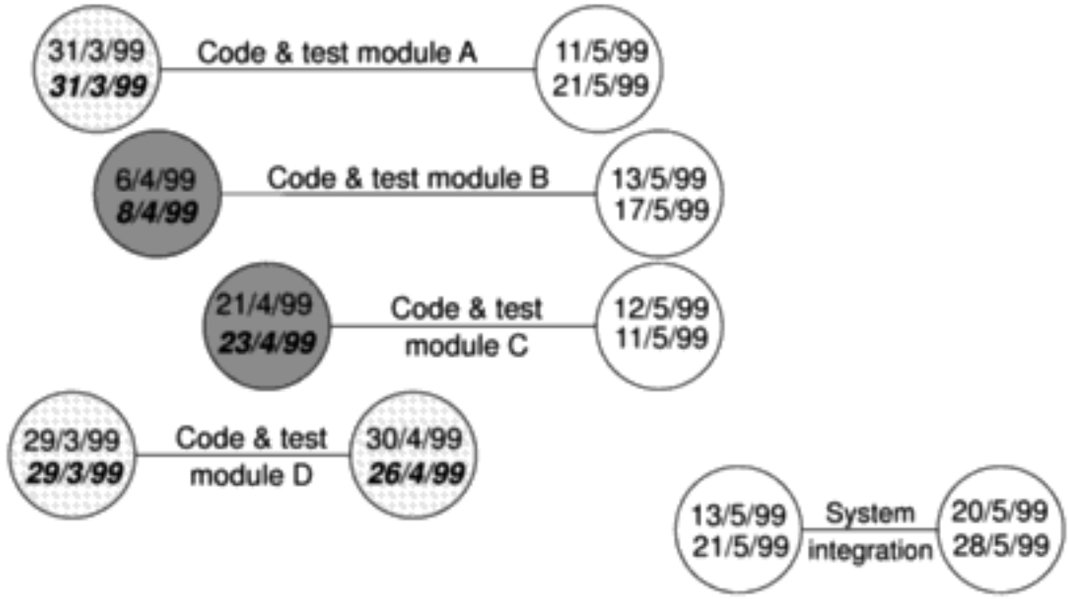


Figure 9.7 *The ball wall chart provides an incentive for meeting targets.*

completion dates – at the start of the project *analyse existing system* is scheduled to be completed by the Tuesday of week 3, *obtain user requirements* by Thursday of week 5, *issue tender*, the final activity, by Tuesday of week 9, and so on.

At the end of the first week Brigette reviews these target dates and leaves them as they are – lines are therefore drawn vertically downwards from the target dates to the end of week one on the actual time axis.

At the end of week two, Brigette decides that *obtain user requirements* will not be completed until Tuesday of week six – she therefore extends that activity line diagonally to reflect this. The other activity completion targets are also delayed correspondingly.

By the Tuesday of week three, *analyse existing system* is completed and Brigette puts a blob on the diagonal timeline to indicate that this has happened. At the end of week three she decides to keep to the existing targets.

At the end of week four she adds another three days to *draft tender* and *issue tender*.

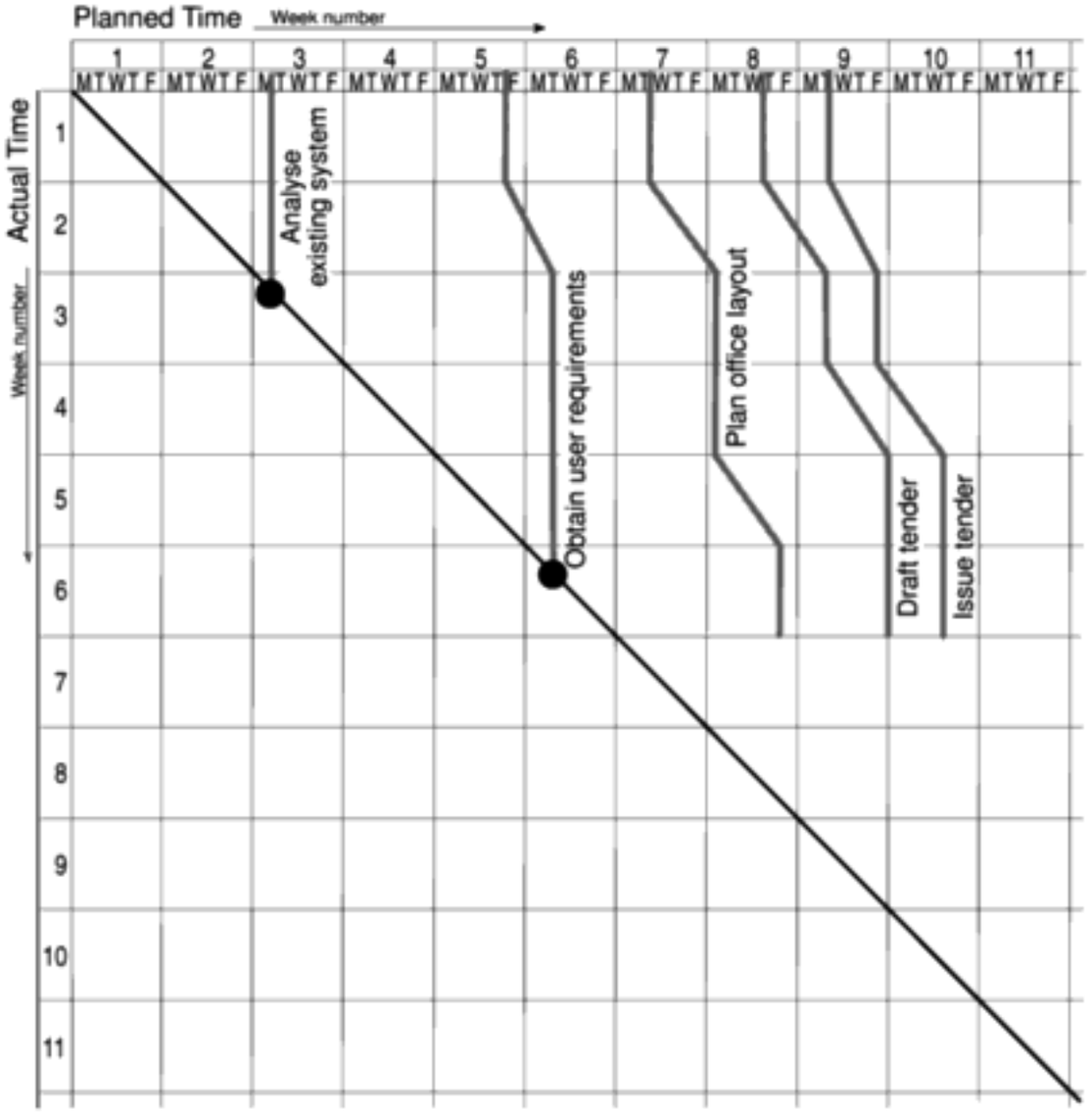
Note that, by the end of week six, two activities have been completed and three are still unfinished. Up to this point she has revised target dates on three occasions and the project as a whole is running seven days late.

Exercise 9.2

By the end of week 8 Brigette has completed planning the office layout but finds that drafting the tender is going to take one week longer that originally anticipated.

What will Brigette's timeline chart look like at the end of week 8?

If the rest of the project goes according to plan, what will Brigette's timeline chart look like when the project is completed?



Brigette's timeline chart contains only the critical activities for her project; ● indicates actual completion of an activity.

For the sake of clarity, the number of activities on a timeline chart must be limited. Using colour helps to distinguish activities, particularly where lines cross.

Figure 9.8 *Brigette's timeline chart at the end of week six.*

The timeline chart is useful both during the execution of a project and as part of the post-implementation review. Analysis of the timeline chart, and the reasons for the changes, can indicate failures in the estimation process or other errors that might, with that knowledge, be avoided in future.

9.5 Cost monitoring

Expenditure monitoring is an important component of project control. Not only in itself, but also because it provides an indication of the effort that has gone into (or at least been charged to) a project. A project might be on time but only because more money has been spent on activities than originally budgeted. A cumulative expenditure chart such as that shown in Figure 9.9 provides a simple method of comparing actual and planned expenditure. By itself it is not particularly

meaningful – Figure 9.9 could, for example, illustrate a project that is running late or one that is on time but has shown substantial costs savings! We need to take account of the current status of the project activities before attempting to interpret the meaning of recorded expenditure.

Project costs may be monitored by a company's accounting system. By themselves, they provide little information about project status.

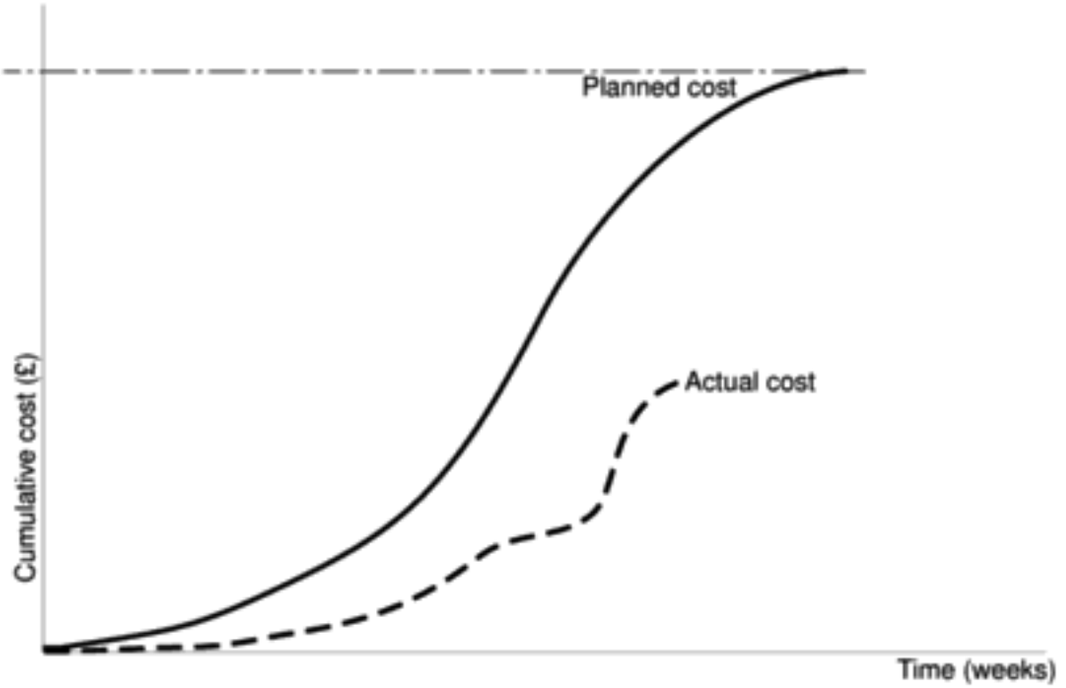


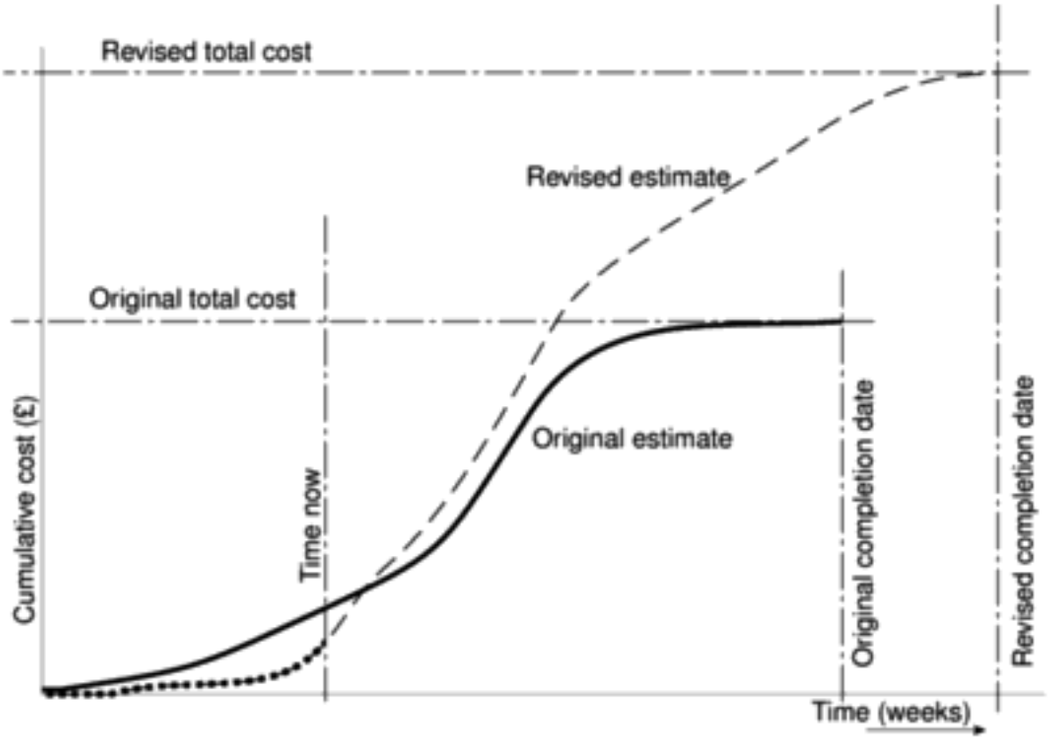
Figure 9.9 *Tracking cumulative expenditure.*

Cost charts become much more useful if we add projected future costs calculated by adding the estimated costs of uncompleted work to the costs already incurred. Where a computer-based planning tool is used, revision of cost schedules is generally provided automatically once actual expenditure has been recorded. Figure 9.10 illustrates the additional information available once the revised cost schedule is included – in this case it is apparent that the project is behind schedule and over budget.

9.6 Earned Value

Earned Value Analysis, also known as Budgeted Cost of Work Performed, is recommended by a number of agencies including the US and Australian departments of defence. It is also recommended in BS 6079.

Earned Value Analysis has gained in popularity in recent years and may be seen as a refinement of the cost monitoring discussed in the previous section. Earned Value Analysis is based on assigning a 'value' to each task or work package (as identified in the WBS) based on the original expenditure forecasts. The assigned value is the original budgeted cost for the item and is known as the *baseline budget* or *budgeted cost of work scheduled* (BCWS). A task that has not started is assigned the value zero and when it has been completed, it, and hence the project, is credited with the value of the task. The total value credited to a project at any point is known as the *earned value* or *budgeted cost of work performed* (BCWP) and this can be represented as a value or as a percentage of the BCWS.



Project costs augmented by project monitoring can be used to generate forecasts of future costs.

Figure 9.10 *The cumulative expenditure chart can also show revised estimates of cost and completion date.*

Where tasks have been started but are not yet complete, some consistent method of assigning an earned value must be applied. Common methods in software projects are:

- **the 0/100 technique** Where a task is assigned a value of zero until such time that it is completed when it is given a value of 100% of the budgeted value;
- **the 50/50 technique** Where a task is assigned a value of 50% of its value as soon as it is started and then given a value of 100% once it is complete;
- **the milestone technique** Where a task is given a value based on the achievement of milestones that have been assigned values as part of the original budget plan.

Of these, we prefer the 0/100 technique. The 50/50 technique can give a false sense of security by over-valuing the reporting of activity starts. The milestone technique might be appropriate for activities with a long duration estimate but, in such cases, it is better to break that activity into a number of smaller ones.

The baseline budget

The first stage in setting up an earned value analysis is to create the *baseline budget*. The baseline budget is based on the project plan and shows the forecast growth in earned value through time. Earned value may be measured in monetary values but, in the case of staff-intensive projects such as software development, it

is common to measure earned value in person-hours or workdays. Amanda's baseline budget, based on the schedule shown in Figure 8.7, is shown in Table 9.2 and diagrammatically in Figure 9.11. Notice that she has based her baseline budget on workdays and is using the 0/100 technique for crediting earned value to the project.

Table 9.2 *Amanda's baseline budget calculation*

Task	*Budgeted workdays*	*Scheduled completion*	*Cumulative workdays*	*% cumulative earned value*
Specify overall system	34	34	34	14.35
Specify module B	15	49	64	27.00
Specify module D	15	49		
Specify module A	20	54	84	35.44
Check specifications	2	56	86	36.28
Design module D	4	60	90	37.97
Design module A	7	63	97	40.93
Design module B	6	66	103	43.46
Check module C spec	1	70	104	43.88
Specify module C	25	74	129	54.43
Design module C	4	79	133	56.12
Code & test module D	25	85	158	66.67
Code & test module A	30	93	188	79.32
Code & test module B	28	94	231	97.47
Code & test module C	15	94		
System integration	6	100	237	100.00

Amanda's project is not expected to be credited with any earned value until day 34, when the activity *specify overall system* is to be completed. This activity was forecast to consume 34 person-days and it will therefore be credited with 34 person-days earned value when it has been completed. The other steps in the baseline budget chart coincide with the scheduled completion dates of other activities.

Monitoring earned value

Having created the baseline budget, the next task is to monitor earned value as the project progresses. This is done by monitoring the completion of tasks (or activity starts and milestone achievements in the case of the other crediting techniques).

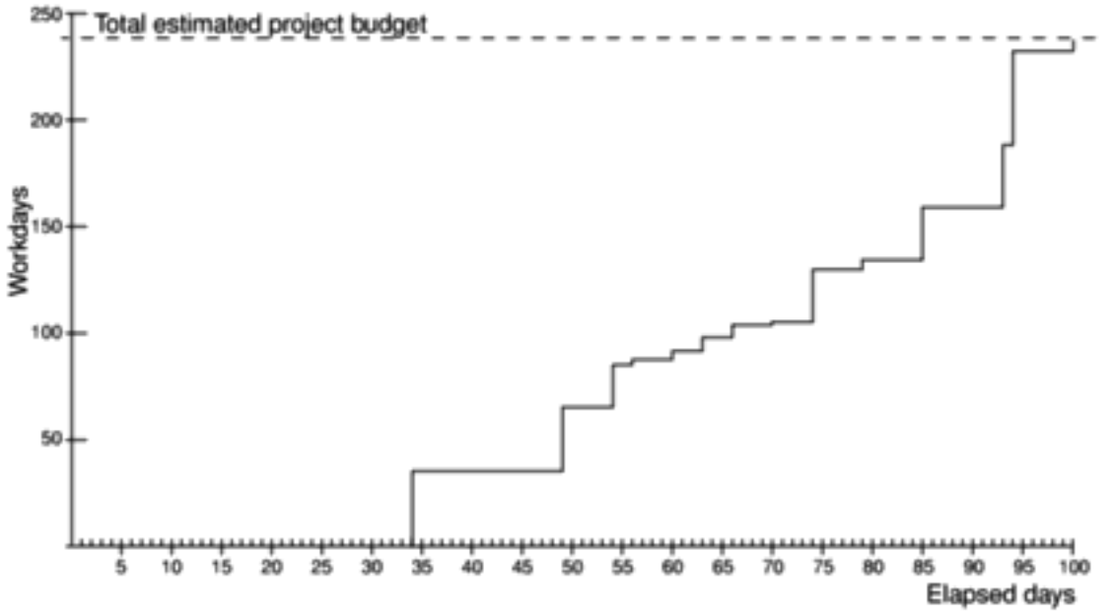


Figure 9.11 *Amanda's baseline budget.*

Exercise 9.3

Figure 9.12 shows Amanda's earned value analysis at the start of week 12 of the project. The earned value (BCWP) is clearly lagging behind the baseline budget, indicating that the project is behind schedule.

By studying Figure 9.12, can you tell exactly what has gone wrong with her project and what the consequences might be?

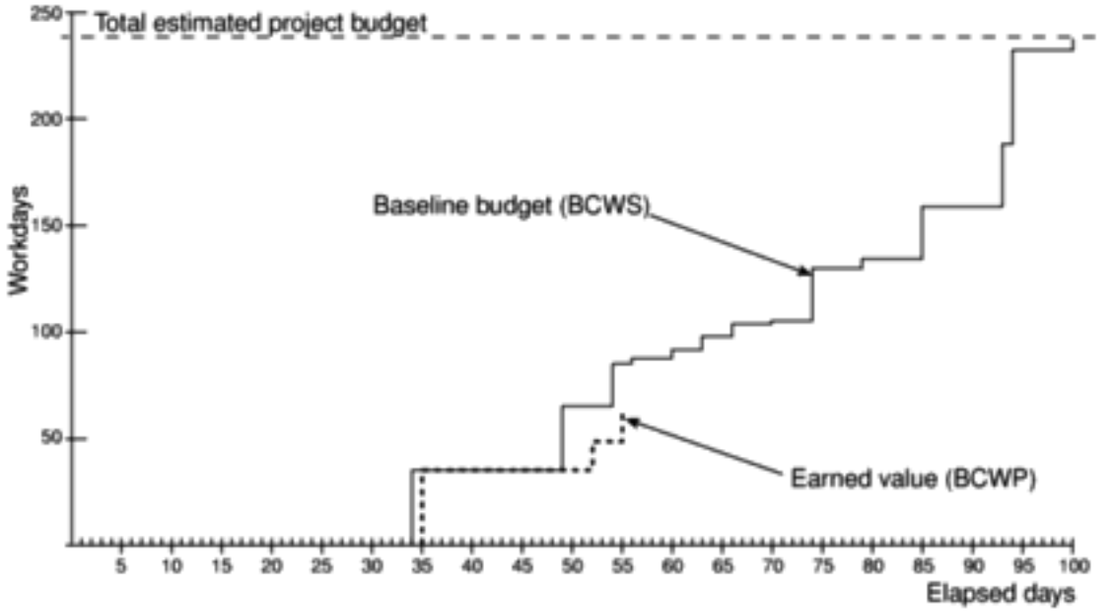


Figure 9.12 *Amanda's earned value analysis at week 12.*

As well as recording BCWP, the actual cost of each task can be collected as *actual cost of work performed*, ACWP. This is shown in Figure 9.13, which, in this case, records the values as percentages of the total budgeted cost.

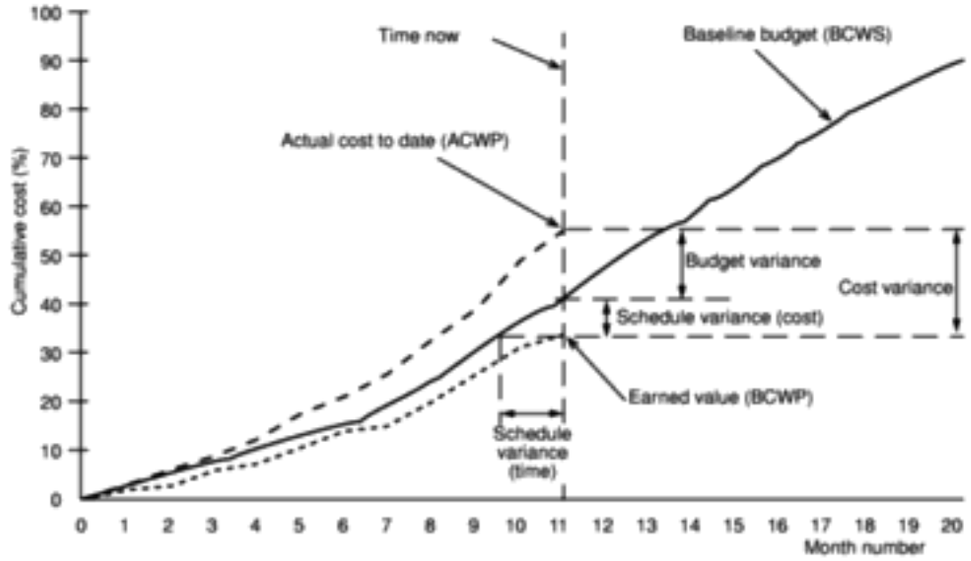


Figure 9.13 *An earned value tracking chart.*

Figure 9.13 also illustrates the following performance statistics, which can be shown directly or derived from the earned value chart.

Budget variance This can be calculated as ACWP – BCWS and indicates the degree to which actual costs differ from those planned.

Schedule variance The schedule variance is measured in cost terms as BCWP – BCWS and indicates the degree to which the value of completed work differs from that planned. Figure 9.13 also indicates the schedule variance in time, which indicates the degree to which the project is behind schedule.

Cost variance This is calculated as BCWP – ACWP and indicates the difference between the budgeted cost and the actual cost of completed work. It is also an indicator of the accuracy of the original cost estimates.

Performance ratios Two ratios are commonly tracked: the *cost performance index* (CPI = BCWP/ACWP) and the *schedule performance index* (SPI = BCWP/BCWS). They can be thought of as a 'value-for-money' indices. A value greater than one indicates that work is being completed better than planned whereas a value of less than one means that work is costing more than and/or proceeding more slowly than planned.

In the same way that the expenditure analysis in Figure 9.9 was augmented to show revised expenditure forecasts, we can augment the simple Earned Value tracking chart with forecasts as illustrated in Figure 9.14.

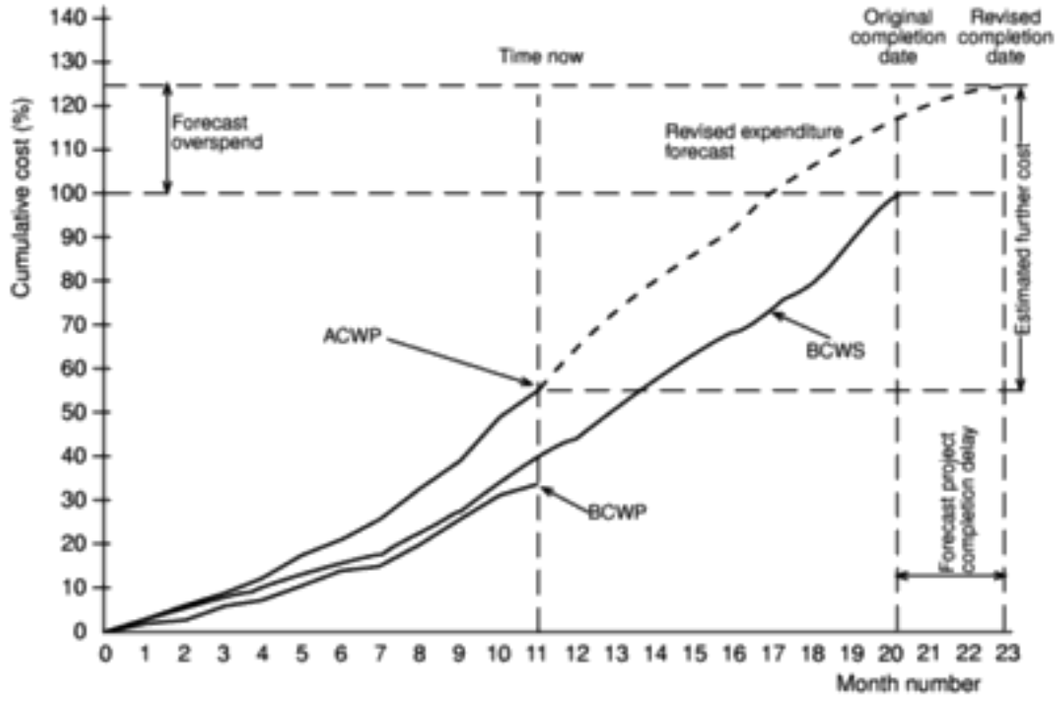


Figure 9.14 *An Earned Value chart with revised forecasts.*

Earned value analysis has not yet gained universal acceptance for use with software development projects, perhaps largely because of the attitude that, whereas a half-built house has a value reflected by the labour and materials that have been used, a half-completed software project has virtually no value at all. This is to misunderstand the purpose of earned value analysis, which, as we have seen, is a method for tracking what has been achieved on a project – measured in terms of the budgeted costs of completed tasks or products.

9.7 Prioritizing monitoring

So far we have assumed that all aspects of a project will receive equal treatment in terms of the degree of monitoring applied. We must not forget, however, that monitoring takes time and uses resources that might sometimes be put to better use!

In this section we list the priorities we might apply in deciding levels of monitoring.

- **Critical path activities** Any delay in an activity on the critical path will cause a delay in the completion date for the project. Critical path activities are therefore likely to have a very high priority for close monitoring.
- **Activities with no free float** A delay in any activity with no free float will delay at least some subsequent activities even though, if the delay is less than the total float, it might not delay the project completion date. These subsequent delays can have serious effects on our resource schedule as a delay in a

Free float is the amount of time an activity may be delayed without affecting any subsequent activity.

subsequent activity could mean that the resources for that activity will become unavailable before that activity is completed because they are committed elsewhere.

- **Activities with less than a specified float** If any activity has very little float it might use up this float before the regular activity monitoring brings the problem to the project manager's attention. It is common practice to monitor closely those activities with less than, say, one week free float.

PERT and the significance of activity duration variance was described in Chapter 7.

- **High risk activities** A set of high risk activities should have been identified as part of the initial risk profiling exercise. If we are using the PERT three-estimate approach we will designate as high risk those activities that have a high estimated duration variance. These activities will be given close attention because they are most likely to overrun or overspend.

- **Activities using critical resources** Activities can be critical because they are very expensive (as in the case of specialized contract programmers). Staff or other resources might be available only for a limited period, especially if they are controlled outside the project team. In any event, an activity that demands a critical resource requires a high level of monitoring.

9.8 Getting the project back to target

A contingency plan should, of course, already exist as a result of the risk analysis methods described in Chapter 7.

Almost any project will, at one time or another, be subject to delays and unexpected events. One of the tasks of the project manager is to recognize when this is happening (or, if possible, about to happen) and, with the minimum delay and disruption to the project team, attempt to mitigate the effects of the problem. In most cases, the project manager tries to ensure that the scheduled project end date remains unaffected. This can be done by shortening remaining activity durations or shortening the overall duration of the remaining project in the ways described in the next section

The schedule is not sacrosanct – it is a plan that should be adhered to so long as it is relevant and cost-effective.

It should be remembered, however, that this might not always be the most appropriate response to disruptions to a plan. There is little point in spending considerable sums in overtime payments in order to speed up a project if the customer is not overly concerned with the delivery date and there is no other valuable work for the team members once this project is completed.

There are two main strategies to consider when drawing up plans to bring a project back on target – shortening the critical path or altering the activity precedence requirements.

Shorten the critical path

The overall duration of a project is determined by the current critical path, so speeding up non-critical path activities will not bring forward a project completion date.

Extolling staff to 'work harder' might have some effect, although frequently a more positive form of action is required, such as increasing the resources available for some critical activity. Fact-finding, for example, might be speeded up by allocating an additional analyst to interviewing users. It is unlikely, however, that the coding of a small module would be shortened by allocating an additional programmer – indeed, it might be counterproductive because of the additional time needed organizing and allocating tasks and communicating.

Resource levels can be increased by making them available for longer. Thus, staff might be asked to work overtime for the duration of an activity and computing resources might be made available at times (such as evenings and week-ends) when they might otherwise be inaccessible.

Where these do not provide a sufficient solution, the project manager might consider allocating more efficient resources to activities on the critical path or swapping resources between critical and non-critical activities. This will be particularly appropriate with staff – an experienced programmer should be significantly more productive than a more junior member of the team.

Time/cost trade-off: there is a general rule that timescales can be shortened by buying more (or more expensive) resources; sometimes this is true.

By such means we can attempt to shorten the timescale for critical activities until such time as either we have brought the project back to schedule or further efforts prove unproductive or not cost-effective. Remember, however, that shortening a critical path often causes some other path, or paths, to become critical (see Section 6.16).

Reconsider the precedence requirements

If attempting to shorten critical activities proves insufficient, the next step is to consider the constraints by which some activities have to be deferred pending completion of others. The original project network would most probably have been drawn up assuming 'ideal' conditions and 'normal' working practices. It might be that, to avoid the project delivering late, it is now worth questioning whether as yet unstarted activities really do have to await the completion of others. It might, in a particular organization, be 'normal' to complete system testing before commencing user training. In order to avoid late completion of a project it might, however, be considered acceptable to alter 'normal' practice and start training earlier.

One way to overcome precedence constraints is to subdivide an activity into a component that can start immediately and one that is still constrained as before. For example, a user handbook can be drawn up in a draft form from the system specification and then be revised later to take account of subsequent changes.

If we do decide to alter the precedence requirements in such a way, it is clearly important to be aware that quality might be compromised and to make a considered decision to compromise quality where needed. It is equally important to assess the degree to which changes in work practices increase risk. It is possible, for example, to start coding a module before its design has been completed. It would normally, however, be considered foolhardy to do so since, as well as compromising quality, it would increase the risk of having to redo some of the

coding once the final design had been completed and thus delay the project even further.

9.9 Change control

So far in this chapter, we have assumed that the nature of the tasks to be carried out has not changed. A project leader like Amanda or Brigette might find, however, that requirements are modified because of changing circumstances or because the users get a clearer idea of what is really needed. The payroll system that Brigette is implementing might, for instance, need to be adjusted if the staffing structure at the college is reorganized.

Other, internal, changes will crop up. Amanda might find that there are inconsistencies in the program specifications that become apparent only when the programs are coded, and these would result in amendments to the specifications.

Careful control of these changes is needed because an alteration in one document often implies changes to other documents and the system products based on that document. The Product Flow Diagrams that have been explained in Chapter 2 indicate relationships between the products of a project where this is the case.

Exercise 9.4

A change in a program specification will normally be carried through into changes to the program design and then changed code. What other products might need to be modified?

Configuration librarian's role

BS EN ISO 9001:1994 (formerly BS 5750) requires that a formal change control procedure be in place.

Control of changes and documentation ought to be the responsibility of someone who may variously be named the Configuration Librarian, the Configuration Manager or Project Librarian. Among this person's duties would be:

- the identification of all items that are subject to change control;
- the establishment and maintenance of a central repository of the master copies of all project documentation and software products;
- the setting up and running of a formal set of procedures to deal with changes;
- the maintenance of records of who has access to which library items and the status of each library item (e.g. whether under development, under test or released).

It will be recalled that it was suggested that the setting up of change control procedures might be one of the first things the Brigette might want to do at Brightmouth College.

Change control procedures

A simple change control procedure for operational systems might have the following steps.

1. One or more users might perceive a need for a modification to a system and ask for a change request to be passed to the development staff.
2. The user management consider the change request and if they approve it pass it to the development management.
3. The development management delegate a member of staff to look at the request and to report on the practicality and cost of carrying out the change. They would, as part of this, assess the products that would be affected by the change.
4. The development management report back to the user management on the findings and the user management decide whether, in view of the cost quoted, they wish to go ahead.
5. One or more developers are authorized to take copies of the master products that are to be modified.
6. The copies are modified. In the case of software components this would involve modifying the code and recompiling and testing it.
7. When the development of new versions of the product has been completed the user management will be notified and copies of the software will be released for user acceptance testing.
8. When the user is satisfied that the products are adequate they will authorize their operational release. The master copies of configuration items will be replaced.

Exercise 9.5

The above steps relate to changes to operational systems. How could they be modified to deal with systems under development?

Changes in scope of a system

A common occurrence with IS development projects is for the size of the system gradually to increase. One cause of this is changes to requirements that are requested by users.

This is sometimes called scope creep.

Exercise 9.6

Think of other reasons why there is a tendency for scope creep.

The scope of a project needs to be carefully monitored and controlled. One way is to re-estimate the system size in terms of SLOC or function points at key milestones.

9.10 Conclusions

In this chapter we have discussed the requirements for the continual monitoring of projects and the need for making progress visible. Among the important points to emerge were:

- planning is pointless unless the execution of the plan is monitored;
- activities that are too long need to be subdivided to make them more controllable;
- ideally, progress should be measured through the delivery of project products;
- progress needs to be shown in a visually striking way, such as through ball charts, in order to communicate information effectively;
- costs need to be monitored as well as elapsed time;
- delayed projects can often be brought back on track by shortening activity times on the critical path or by relaxing some of the precedence constraints.

9.11 Further exercises

1. Take a look at Amanda's project schedule shown in Figure 8.7. Identify those activities scheduled to last more than three weeks and describe how she might monitor progress on each of them on a fortnightly or weekly basis.
2. Amanda's Gantt chart at the end of week 17 (Figure 9.5) indicates that two activities are running late. What effect might this have on the rest of the project? How might Amanda mitigate the effects of this delay?
3. Table 9.2 illustrates Amanda's earned value calculations based on workdays. Revise the table using monetary values based on the cost figures that you used in Exercise 8.5. Think carefully about how to handle the costs of Amanda as project manager and the recovered overheads and justify your decisions about how you treat them.
4. If you have access to project planning software, investigate the extent to which it offers support for earned value analysis. If it does not do so directly, investigate ways in which it would help you to generate a baseline budget (BCWS) and track the earned value (BCWP).
5. Describe a set of change control procedures that would be appropriate for Brigette to implement at Brightmouth College.

Chapter 10

Managing contracts

OBJECTIVES

When you have completed this chapter, you will be able to:

- understand the advantages and disadvantages of using goods and services brought in from outside the organization;
- distinguish among the different types of contract;
- follow the stages needed to negotiate an appropriate contract;
- outline the contents of a contract for goods and services;
- plan the evaluation of a proposal or product;
- administer a contract from its signing until the final acceptance of project completion.

10.1 Introduction

In the Brightmouth College scenario, the management of the college have made a decision to obtain their software from an external supplier. Given the range of payroll software on the market and their own limited capability for developing new and reliable software, this would seem sensible. Meanwhile at IOE, Amanda has available, at least in theory, a team of software developers who are employees of IOE. However, the demand for software design and construction effort will fluctuate, rising when a new project is initiated and trailing off as it is completed. In-house developers could thus have periods of intense pressure when new projects are being developed, interspersed by periods of relative idleness. The IOE management might therefore decide that it would be more cost-effective to get an outside software house to carry out the new development while a reduced group of in-house software development staff remain busy maintaining and giving support to the users of existing systems.

It is not unusual for a major organization to spend 6 to 12 months and 40% of the total acquisition and implementation budget on package evaluation with major customer service and support applications (Demian Martinez, Decision Drivers Inc., *Computing*, 23 July 1998).

The buying in of both goods and services, rather than 'doing it yourself', is attractive when money is available but other, less flexible, types of resource, especially staff time, are in short supply. However, there are hazards for organizations who adopt this policy. Many of these potential dangers arise from the fact that considerable staff time and attention will still be needed to manage a contracted out project successfully. Although the original motivation for contracting out might have been to reduce management effort, it is essential that customer organizations such as Brightmouth College and IOE find time to make clear their exact requirements at the beginning of the planned work, and also to ensure that the goods and services that result are in fact what are actually required.

Also, it needs hardly be said that potential suppliers are more likely to be flexible and accommodating before any contract has been signed than they will be afterwards – especially if the contract is for a fixed price. All this points to the need for as much forethought and planning with an acquisition project as with an internal development project.

In the remainder of this chapter, we will first discuss the different types of contract that can be negotiated. We will then follow through the general steps that ought to be followed when placing a contract. The issues that ought to be considered when drafting a contract are then examined. We conclude by describing some of the things that need to be done while the contract is actually being executed.

It was, for example, reported that two consortia led by Sema and EDS respectively had spent £4 million over two years bidding for a UK government project to renew the IT infrastructure in the prison service – the final job was estimated as being worth £350 million (*Computing*, 13 August, 1998).

Note that the bargaining position of the customer will be much stronger if their business is going to be very valuable. If you are buying a cut-price computer game from a local store, you are unlikely to be able to negotiate variations on the supplier's standard contract of sale! (Indeed, because of the inequality of the parties in such circumstances, such sales are subject to special consumer protection laws). It is reasonable for potential suppliers to weigh up carefully the time and money they are willing to spend responding to a customer's initial request, as there is no guarantee of their obtaining the final contract.

10.2 Types of contract

The external resources required could be in the form of *services*. A simple example of this could be using temporary staff on short term contracts to carry out some project tasks. At Brightmouth College, Brigette could use temporary staff to type into the computer system the personnel details needed to set up the payroll standing data for the new system, while at IOE a decision might be made to carry out the required system building in-house but to augment the permanent staff with contract programmers for the duration of the project. A more far-reaching use of external services would be for the contractor not only to supply the new system but to also operate it on the customer's behalf. For example, it might well be worth Brightmouth College abandoning the idea of buying a package and instead getting a payroll services agency to carry out all the payroll work on their behalf.

On the other hand, the contract could be placed for the supply of a *completed software application*.

This could be:

- a *bespoke* system, that is, a system that is created from scratch specifically for one customer;
- *off-the-shelf*, which you buy 'as is' – this is sometimes referred to as *shrink-wrapped* software;
- *customized off-the-shelf* (COTS) software – this is a basic core system, which is modified to meet the needs of a particular customer.

Where equipment is being supplied then, in English law, this may be regarded as a contract for the supply of *goods*. In the case of the supply of software this may be regarded as supplying a service (to write the software) or the granting of a *licence* (or permission) to use the software, which remains in the ownership of the supplier. These distinctions will have legal implications.

David Bainbridge's *Introduction to Computer Law*, Pitman, 3e, 1996 is highly recommended as a guide to the legal aspects of IT contracts.

Exercise 10.1

Which of the three system options (that is, bespoke, off-the-shelf or COTS) might Amanda consider with regard to the IOE maintenance group accounts system? What factors would she need to take into account?

Another way of classifying contracts is by the way that the payment to suppliers is calculated. We will look at:

- fixed price contracts;
- time and materials contracts;
- fixed price per delivered unit contracts

The section on ways of assessing supplier payments draws heavily on material from Paul Radford and Robyn Lawrie of Charismatek Software Metrics, Melbourne, Australia.

Fixed price contracts

As the name implies, in this situation a price is fixed when the contract is signed. The customer knows that, if there are no changes in the contract terms, this is the price to be paid on the completion of the work. In order for this to be effective, the customer's requirement has to be known and fixed at the outset. In other words, when the contract is to construct a software system, the detailed requirements analysis must already have been carried out. Once the development is under way, the customer will not be able to change their requirements without renegotiating the price of the contract.

The advantages of this method are the following.

- **Known customer expenditure** If there are few subsequent changes to the original requirements, then the customer will have a known outlay.
- **Supplier motivation** The supplier has a motivation to manage the delivery of the system in a cost-effective manner.

The disadvantages include the following.

- **Higher prices to allow for contingency** The supplier absorbs the risk for any errors in the original estimate of product size. To reduce the impact of this risk, the supplier will add a margin when calculating the price to be quoted in a tender.
- **Difficulties in modifying requirements** The need to change the scope of the requirements sometimes becomes apparent as the system is developed – this can cause friction between the supplier and customer.
- **Upward pressure on the cost of changes** When competing against other potential suppliers, the supplier will try and quote as low a price as possible. If, once the contract is signed, further requirements are put forward, the supplier is in a strong position to demand a high price for these changes.
- **Threat to system quality** The need to meet a fixed price can mean that the quality of the software suffers.

Time and materials contracts

With this type of contract, the customer is charged at a fixed rate per unit of effort, for example, per staff-hour. At the start of the project, the supplier normally provides an estimate of the overall cost based on their current understanding of the customer's requirements, but this is not the basis for the final payment.

The advantages of this approach are the following.

- **Ease of changing requirements** Changes to requirements are dealt with easily. Where a project has a research orientation and the direction of the project changes as options are explored, then this can be an appropriate method of calculating payment.
- **Lack of price pressure** The lack of price pressure can allow better quality software to be produced.

The disadvantages of this approach are:

- **Customer liability** The customer absorbs all the risks associated with poorly defined or changing requirements.
- **Lack of incentives for supplier** The supplier has no incentive to work in a cost-effective manner or to control the scope of the system to be delivered.

Because the supplier appears to be given a blank cheque, this approach does not normally find favour with customers. However, the employment of contract development staff, in effect, involves this type of contract.

Fixed price per unit delivered contracts

Function point counting was discussed in Chapter 5.

This is often associated with function point (FP) counting. The size of the system to be delivered is calculated or estimated at the outset of the project. The size of

the system to be delivered might be estimated in lines of code, but FPs can be more easily and reliably derived from requirements documents. A price per unit is also quoted. The final price is then the unit price multiplied by the number of units. Table 10.1 shows a typical schedule of prices.

Table 10.1 *A schedule of charges per function point*

Function point count	*Function design cost per FP*	*Implementation cost per FP*	*Total cost per FP*
Up to 2,000	$242	$725	$967
2,001–2,500	$255	$764	$1,019
2,501–3,000	$265	$793	$1,058
3,001–3,500	$274	$820	$1,094
3,501–4,000	$284	$850	$1,134

This Table comes from D. Garmus and D. Herron, *Measuring The Software Process*, Prentice Hall, 1996.

The company who produced this table, RDI Technologies of the USA, in fact charge a higher fee per FP for larger systems. For example, a system to be implemented contains 2,600 FPs. The overall charge would be 2,000 × $967, plus 500 × $1,1019, plus 100 × $1,058.

One problem that has already been noted is that the scope of the system to be delivered can grow during development. The software supplier might first carry out the system design. From this design, an FP count could be derived. A charge could then be made for design work based on the figures in the 'Function design cost per FP' column. This, if the designed system was counted at 1,000 FPs, would be 1000 × $242 = $242,000. If the design were implemented, and the actual software constructed and delivered, then the additional 1000 × $725 = $725,000 would be charged. If the scope of the system grows because the users find new requirements, these new requirements would be charged at the combined rate for design and implementation. For example, if new requirements amounting to 100 extra FPs were found, then the charge for this extra work would be $967 × 100 = $96,700.

Exercise 10.2

A system to be designed and implemented is counted as comprising 3,200 FPs. What would be the total charge according to the schedule in Table 10.1?

The advantages of this approach are as follows.

- **Customer understanding** The customer can see how the price is calculated and how it will vary with changed requirements.
- **Comparability** Pricing schedules can be compared.
- **Emerging functionality** The supplier does not bear the risk of increasing functionality.

- **Supplier efficiency** The supplier still has an incentive to deliver the required functionality in a cost-effective manner (unlike with time and materials contracts).
- **Life cycle range** The requirements do not have to be definitively specified at the outset. Thus the development contract can cover both the analysis and design stages of the project.

The disadvantages of this approach are as follows.

The impact of late changes will be further discussed in Chapter 12 on Software Quality.

- **Difficulties with software size measurement** Lines of code can easily be inflated by adopting a verbose coding style. With FPs, there can be disagreements about what the FP count should really be: in some cases, FP counting rules might be seen as unfairly favouring either the supplier or customer. Users, in particular, will almost certainly not be familiar with the concept of FPs and special training might be needed for them. The solution to these problems might be to employ an independent FP counter.
- **Changing requirements** Some requested changes might affect existing transactions drastically but not add to the overall FP count. A decision has to be taken about how to deal with these changes. A change made late in the development cycle will almost certainly require more effort to implement than one made earlier.

To reduce the last difficulty, one suggestion from Australia has been to vary the charge depending on the point at which they have been requested – see Table 10.2.

The table comes from the draft Acquisition of customized software policy document, published by the Department of State Development, Victoria, 1996.

Table 10.2 *Examples of additional charges for changed functionality*

	Pre-acceptance testing handover	*Post-acceptance testing handover*
Additional FPs	100%	100%
Changed FPs	130%	150%
Deleted FPs	25%	50%

Exercise 10.3

A contract stipulates that a computer application is to be designed, constructed and delivered at a cost of $600 per FP. After acceptance testing, the customer asks for changes to some of the functions in the system amounting to 500 FPs and some new functions which amount to 200 additional FPs. Using Table 10.2, calculate the additional charge.

In addition to the three payment methods above, there are other options and permutations of options. For instance, the implementation of an agreed specification could be at a fixed price, with provision for any additions or changes to the requirements to be charged for on a per FP basis. Another example could be where the contractor has to buy in large amounts of equipment, the price of which

might fluctuate through no fault of the contractor. In this case it is possible to negotiate a contract where the final price contains a fixed portion for labour plus an amount that depends on the actual cost of a particular component used.

Exercise 10.4

It is easy to see why passing on fluctuations in equipment costs can be advantageous to the contractor. However, is there any advantage to the customer in such an arrangement?

An underlying problem with software can once again be seen when the questions of contractual obligations and payment are considered – namely its relative invisibility and its flexibility. These mean that system size and consequently development effort tends to be very difficult to judge. If contractors are realistically to quote firm prices for work, then the tasks they are asked to undertake must be carefully constrained. For example, it would be unrealistic for a contractor to be asked to quote a single price for all the stages of a development project: how can they estimate the construction effort needed when the requirements are not yet established? For this reason it will often be necessary to negotiate a series of contracts, each covering a different part of the system development life cycle.

The concept of 'IS adaptations' in Euromethod (see Appendix C) supports the idea of a series of separate contracts to implement a single system.

Another way of categorizing contracts, at least initially, is according to the approach that is used in contractor selection:

- open;
- restricted;
- negotiated.

This categorization is based on European Union regulations.

Open tendering process

In this case, any supplier can bid to supply the goods and services. Furthermore all bids that are compliant with the original conditions laid down in the *invitation to tender* must be considered and evaluated in the same way as all the others. With a major project where there are lots of bids and the evaluation process is time-consuming, this can be an expensive way of doing things.

In recent years, there has been a global movement towards removing artificial barriers that hamper businesses in one country supplying goods and services in another. Examples of this are the drives by bodies such GATT and the European Union to ensure that national governments and public bodies do not limit the granting of contracts to their fellow nationals without good reason. One element of this is the laying down of rules about how tendering processes should be carried out. In certain circumstances, these demand that an open tendering process be adopted.

GATT stands for 'General Agreement on Trade and Tariffs'. The specific part of GATT that is relevant here is the Agreement on Government Procurement. GATT covers the EU and also several other countries including the US and Japan.

Restricted tendering process

In this case, there are bids only from suppliers who have been invited by the customer. Unlike the open tendering process, the customer may at any point reduce the number of potential suppliers being considered. This is usually the best approach to be adopted. However, it is not without risk: where the resulting contract is at a fixed price, the customer assumes responsibility for the correctness and completeness of the requirements specified to the prospective suppliers. Defects in this requirements documentation sometimes allow a successful bidder subsequently to claim for additional payments.

Negotiated procedure

There are often, however, some good reasons why the restricted tendering process might not be the most suitable in some particular sets of circumstances. Say, for example, that there is a fire that destroys part of an office, including IT equipment. The key concern here is to get replacement equipment up and running as quickly as possible and there is no time to embark on a lengthy tendering process. Another situation might be where a new software application had been successfully built and implemented by an outsider, but the customer decides to have some extensions to the system. As the original supplier has staff who have complete familiarity with the existing system, it might once again be inconvenient to approach other potential suppliers via a full tendering process.

In these cases, an approach to a single supplier might be justified. However, it takes little imagination to realise that approaching a single supplier can open the customer up to charges of favouritism and should be done only where there is a clear justification.

10.3 Stages in contract placement

We are now going to discuss the typical stages in awarding a contract.

Requirements analysis

This discussion assumes that a feasibility study has already provisionally identified the need for the intended software.

Before potential supplier can be approached, you need to have a clear set of requirements. Two points need to be emphasized here. The first is that it is easy for this step to be skimped where the user has many day-to-day pressures and not much time to think about future developments. In this situation, it can be useful to bring in an external consultant to draw up a requirements document. Even here, users and their managers need to look carefully at the resulting requirements document to ensure that it accurately reflect their needs. As David Bainbridge has pointed out: '*the lack of, or defects in, the specification are probably the heart of most disputes resulting from the acquisition of computer equipment and software*'

David Bainbridge *ibid* page 135.

The requirements document might typically have sections with the headings shown in Table 10.3. This requirements document is sometimes called an operational requirement or OR.

Table 10.3 *Main sections in a requirements document*

	Section name
1	Introduction
2	A description of any existing systems and the current environment
3	The customer's future strategy or plans
4	System requirements • mandatory • desirable
5	Deadlines
6	Additional information required from potential suppliers

The requirements define carefully the *functions* that need to be carried out by the new application and all the necessary *inputs* and *outputs* for these functions. The requirements should also state any *standards* with which there should be compliance, and the existing systems with which the new system needs to be compatible. As well as these functional requirements, there will also need to be operational and quality requirements concerning such matters as the required response times, reliability, usability and maintainability of the new system.

Chapter 12 on Software Quality discusses how aspects of quality can be measured.

In general, the requirements document should state *needs* as accurately as possible and should avoid technical specifications of possible solutions. The onus should be placed on the potential suppliers to identify the technical solutions that they believe will meet the customer's stated needs. After all, they are the technical experts who should have access to the most up-to-date information about current technology.

Each requirement needs to be identified as being either *mandatory* or *desirable*.

- **Mandatory** If a proposal does not meet this requirement then the proposal is to be immediately rejected. No further evaluation would be required.
- **Desirable** A proposal might be deficient in this respect, but other features of the proposal could compensate for it.

For example, in the case of the Brightmouth College payroll acquisition project, Brigette might identify as a mandatory requirement that any new system should be able to carry out all the processes previously carried out by the old system. However a desirable feature might be that the new payroll software should be able to produce accounting details of staff costs in an electronic format that can be read directly by the college's accounting computer system.

One suggestion is that the weighting between product criteria and supplier criteria when selecting software ought to be 50:50 (Demian Martinez, Decision Drivers Inc., *Computing* 23 July 1998).

Among the other details that should be included in the requirements document to be issued to potential suppliers would be requests for any information needed to help us judge the standing of organization itself. This could include financial reports, references from past customers and the CVs of key development staff.

Evaluation plan

Having drawn up a list of requirements, we now need to draw up a plan of how the proposals that are submitted are to be evaluated. The situation will be slightly different if the contract is for a system that is to be specially written as opposed to an off-the-shelf application. In the latter case, it is the system itself that is being evaluated while in the former situation it is a proposal for a system.

First, a means of checking that all the mandatory requirements have been met needs to be identified. The next consideration is of how the desirable requirements can be evaluated. The problem here is weighing the value of one quality against another. The ISO 9126 standard, which is discussed in the chapter on software quality, can be used to decide that one system has more 'quality' than another, but if there is a difference in price between the two, we need to be able to estimate if the increase in quality is worth the additional price. Hence 'value for money' is often the key criterion. For example, we mentioned above an instance where the existence of an accounting link file was identified as a desirable requirement in the case of the Brightmouth College payroll acquisition project. Could a financial value be placed on this? If we were to cost clerical effort at £20 an hour and we knew that four hours of clerical effort a month went into entering staffing costs into the accounting computer system, then we could conclude that over a four year period (£20 an hour × 4 hours a month × 48 months), or £3840, would be saved. If system A has this feature and costs £1000 more than system B, which does not, then this would seem to give system A an advantage. If, however, system A cost £5000 more than B then the picture would be different.

Some of these issues were touched on in Chapter 3 on project evaluation.

It needs to be stressed that the costs to be taken into account are those for the whole of the lifetime of the proposed system, not just the costs of acquiring the system. Also, where the relationship with the supplier is likely to be ongoing, the supplier organization needs to be assessed as well as its products.

Exercise 10.5

One desirable feature sought in the Brightmouth College payroll is the ability to raise staff to the next point in their salary scale automatically at the beginning of each payroll year. At present, the new scale points have to be input clerically and then be checked carefully. This takes about 20 hours of staff effort each year, which can be costed at £20 an hour. System X has this feature, but system Y does not. System X also has a feature which can automatically produce bar-charts showing payroll expenditure per department. Such a report currently has to be produced twice a year by hand and on each occasion takes about 12 hours effort to complete. With System Y, changes to department names can be carried out without any coding effort, whereas in the case of System X, the supplier would charge a minimum of £300 to do this. The college authorities estimate that there is 50% chance that this could occur during the expected four year lifetime of the system. System X costs £500 more than System Y. On the basis of this information which system appears to give better value for money?

Invitation to tender

Having produced the requirements and the evaluation plan, it is now possible to issue the invitation to tender to prospective suppliers. Essentially, this will be the requirement document with a supporting letter, which may have additional information about how responses to the invitation are to be lodged. A deadline will be specified and it is hoped that by then a number of proposals with price quotations will have been received.

In English law, for a contract to exist there must be an offer on one side that must be accepted by the other side. The invitation to tender is not an offer itself but an invitation for prospective suppliers to make an offer.

Certain new problems now emerge. The requirements that have been laid down might be capable of being satisfied in a number of different ways. The customer needs to know not only a potential supplier's price but also the way in which they intend to satisfy the requirements – this will be particularly important where the contract is to build a new system from scratch.

In some relatively straight-forward cases, it would be enough to have some post-tender clarification and negotiation to resolve issues in the supplier's proposal. With more complex projects a more elaborate approach may be needed. One way of getting the detail of the suppliers' proposals elaborated is to have a two stage tendering process.

This approach is recommended by the CCTA in the United Kingdom.

In the first stage, technical proposals are requested from potential suppliers who do not necessarily quote any prices at this stage. Some of these proposals can be dismissed out of hand as not being able to meet the mandatory requirements. With the remaining ones, discussions may be held with representatives of the suppliers in order to clarify and validate the technical proposals. The suppliers may be asked to demonstrate certain aspects of their proposals. Where short-comings in the proposal are detected, the supplier may be given the opportunity to remedy these.

The result of these discussions could be a *Memorandum of Agreement* (MoA) with each prospective supplier. This is an acceptance by the customer that the proposed solution (which might have been modified during discussions) offered by the supplier satisfactorily meets the customer's requirement.

In the second stage, tenders are invited from all the suppliers who have signed individual Memoranda of Agreement. The tender would incorporate the MoA and would be concerned with the financial terms of a potential contract.

If a design has to be produced as part of the proposal made by a supplier in response to an invitation to tender, then a difficulty would be that the supplier would have to do a considerable amount of detailed design work with only a limited prospect of being paid for it. One way of reducing this burden is for the customer to choose a small number of likely candidates who will be paid a fee to produce design proposals. These can then be compared and the final contract for construction awarded to the most attractive proposal.

The ISO 12207 has a rather different approach. Once a contract for software construction is signed, the supplier develops a design, which then has to be agreed by the customer.

Evaluation of proposals

This needs to be done in a methodical and planned manner. We have already mentioned the need to produce an evaluation plan, which will describe how each proposal will be checked to see if it meets each requirement. This reduces risks of requirements being missed and ensures that all proposals are treated consistently. Otherwise, there is a risk that a proposal might be unfairly favoured because of the presence of a feature that was not requested in the original requirement.

It will be recalled that an application could be either bespoke, off-the-shelf, or customized. In the case of off-the-shelf packages, it would be the software itself that would be evaluated and it might be possible to combine some of the evaluation with acceptance testing. With bespoke development it would be a proposal that is evaluated, while COTS may involve elements of both. Thus different planned approaches would be needed in each case.

The process of evaluation may include:

- scrutiny of the proposal documents;
- interviewing suppliers' representatives;
- demonstrations;
- site visits;
- practical tests.

The proposal documents provided by the suppliers can be scrutinized to see if they contain features satisfying all the original requirements. Clarification might need to be sought over certain points. Any factual statements made by a supplier imply a legal commitment on their part if it influences the customer to offer the contract to that supplier. It is therefore important to get a written, agreed, record of these clarifications. The customer may take the initiative here by keeping notes of meetings and then writing afterwards to the suppliers to get them to confirm the accuracy of the notes. A supplier might, in the final contract document, attempt to exclude any commitment to any representations that have been made in pre-contract negotiations – the terms of the contract need to be scrutinized in this respect.

Where the delivered product is to be based on an existing product it might be possible to see a demonstration. A danger with demonstrations is that they can be controlled by the supplier and as a passive observer it is often difficult to maintain full attention for more than, say, half-an-hour. Because of this, the customer should produce a schedule of what needs to be demonstrated, ensuring that all the important features are seen in operation.

With off-the-shelf software, it should be possible to have actual access to the application. For example, a demonstration version, which closes itself down after 30 days, might be available. Once again a test plan is needed to ensure that all the important features are evaluated in a complete and consistent manner. Once a particular package emerges as the most likely candidate, it needs to be carefully

investigated to see if there are any previously unforeseen factors that might invalidate this choice.

A frequent problem is that while an existing application works well on one platform with a certain level of transactions, it does not work satisfactorily on the platform that the customer has or at the level of throughput that it would be subjected to in the customer's work environment. Demonstrations will not generally reveal this problem. Visits to operational sites already using the system will be more informative in this respect. In the last resort a special volume test could be conducted.

Exercise 10.6

How would you evaluate the following aspects of a proposal?

i. The usability of an existing software application.

ii. The usability of a software application that is yet to be designed and constructed.

iii. The maintenance costs of hardware to be supplied.

iv. The time taken to respond to requests for software support.

v. Training.

Where substantial sums of money are involved, legal advice on the terms of the contract is essential.

Eventually a decision will be made to award the contract to one of the suppliers. One of the central reasons for using a structured and, as far as possible, objective approach to evaluation is to be able to demonstrate that the decision has been made impartially and on merit. In most large organizations, placing a contract involves the participation of a second party within the organization, such as a contracts department, who can check that the correct procedures have been carried out. Also the final legal format of a contract will almost certainly require some legal expertise.

In any case, not only should the successful candidate be notified but the unsuccessful candidates should also be told of the decision. This is not simply a matter of courtesy: under GATT or EU rules, there is a legal requirement to do this in certain circumstances. It makes dealing with unsuccessful bidders easier if they can be given clear and objective reasons why their proposals did not find favour.

10.4 Typical terms of a contract

In a textbook such as this, it is not possible to describe all the necessary content of contracts for IT goods or services. It is possible, however to outline some of the major areas of concern.

Definitions

The terminology used in the contract document may need to be defined, for example, who is meant by the words 'client' and 'supplier'.

Form of agreement

For example, is it a contact of sale, a lease, or a licence? Also, can the subject of the contract, such as a licence to use a software application, be transferred to another party?

Goods and services to be supplied

Equipment and software to be supplied This includes an actual list of the individual pieces of equipment to be delivered, complete with the specific model numbers.

Services to be provided This covers such things as:

- training;
- documentation;
- installation;
- conversion of existing files;
- maintenance agreements;
- transitional insurance arrangements.

Ownership of the software

Who has ownership of the software? There are two key issues here: firstly, whether the customer can sell the software to others and, secondly, whether the supplier can sell the software to others. Where off-the-shelf software is concerned, the supplier often simply grants a license for you to use the software. Where the software is being written specially for a customer, then that customer will normally wish to ensure exclusive use of the software – they may object to software which they hoped would give them a competitive edge being sold to their rivals. They could do this by acquiring the copyright to the software outright or by specifying that they should have *exclusive use* of the software. This would need to be written into the contract. Where a core system has been customized by a supplier, then there is less scope for the customer to insist on exclusive use.

Where software is written by an employee as part of a contract of employment, it is assumed that the copyright belongs to the employer. Where the customer organization has contracted an external supplier to write software, the contract needs to make clear who is going to retain the copyright – it cannot, in this case, be automatically assumed it is the customer. The customer might have decided to take over responsibility for maintenance and further development once the software is delivered and in this case will need access to the source code. In other

cases, where the customer does not have an adequate in-house maintenance function, the supplier can retain the source code, and the customer will have to approach the supplier for any further changes. There are many potential dangers with this, not the least being that the supplier could go out of business. An escrow agreement can be included in the contract so that up-to-date copies of the source code are deposited with a third party. In the United Kingdom, the National Computing Centre provide an escrow service.

Environment

Where physical equipment is to be installed, the demarcation line between the supplier's and customer's responsibilities with regard to such matters as accommodation and electrical supply needs to be specified. Where software is being supplied, the compatibility of the software with the existing hardware and operating system platforms would need to be confirmed.

Customer commitments

Even when work is carried out by external contractors, a development project still needs the participation of the customer. The customer will have to provide accommodation for the suppliers and perhaps other facilities such as telephone lines.

Acceptance procedures

Good practice would be to accept a delivered system only after it has undergone user acceptance tests. This part of the contract would specify such details as the time that the customer will have to conduct the tests, deliverables upon which the acceptance tests depend and the procedure for signing off the testing as completed.

Some customers find that specially written or modified software is not thoroughly tested by the supplier before delivery. Some suppliers seem to think that it is cheaper to get the customer to do the testing for them!

Standards

This covers the standards with which the goods and services should comply. For example, a customer can require the supplier to conform to the ISO 12207 standard relating to the software life cycle and its documentation (or, more likely, a customized sub-set of the standard). Within the European Union, government customers with contracts for projects above a certain threshold value must, by law, ensure that the work conforms to certain standards.

Project and quality management

The arrangements for the management of the project must be agreed. Among these would be frequency and nature of progress meetings and the progress information to be supplied to the customer. The contract could require that appropriate ISO 9000-series standards be followed. The ISO 12207 standard provides for the customer to have access to quality documentation generated internally by the supplier, so that the customer can ensure that there is adherence to standards.

Timetable

This provides a schedule of when the key parts of the project should be completed. This timetable will commit both the supplier and the customer. For example, the supplier might be able to install the software on the agreed date only if the customer makes the hardware platform available at that point.

Price and payment method

Obviously the price is very important! What also needs to be agreed is when the payments are to be made. The supplier's desire to be able to meet costs as they are incurred needs to be balanced by the customer's requirement to ensure that goods and services are satisfactory before parting with their money.

Miscellaneous legal requirements

This is the legal small print. Contracts often have clauses that deal with such matters the legal jurisdiction that will apply to the contract, what conditions would apply to the sub-contracting of the work, liability for damage to third parties, and liquidated damages. *Liquidated damages* are estimates of the financial losses that the customer would suffer if the supplier were to fall short of their obligations. It is worth noting that under English law, the penalties laid down in penalty clauses must reflect the actual losses the customer would suffer and cannot be unrealistic and merely punitive. Even this limitation will not be enough in some cases as far as the supplier is concerned. As computer systems assume increasingly critical roles in many organizations and in safety-critical systems can even be life-threatening in the case of malfunction, the possible consequential damage could be very great. Suppliers will not unnaturally try to limit this kind of liability. The courts (in England and Wales) have tended to look critically at such attempts at limiting liability, so that suppliers will, in the case of major contracts, take out insurance to cover such liabilities.

If there is a dispute, resorting to litigation, while being lucrative to the lawyers involved, is both time-consuming and expensive. An alternative is to agree that disputes be settled by *arbitration*. This requires that any dispute be referred to an expert third party whose decision as to the facts of the case is binding. Even this procedure is seldom quick and inexpensive and another option is *alternative dispute resolution* where a third party acts as a mediator who has only an advisory capacity and attempts to broker an agreement between the two sides.

10.5 Contract management

Euromethod offers guidance about how decision points can be planned.

We now need to consider the communications between the supplier and the customer while the work contracted for is being carried out. It would probably suit all concerned if the contractor could be left to get on with the work undisturbed. However, at certain *decision points*, the customer needs to examine work already done and make decisions about the future direction of the project. The project will

require representatives of the supplier and customer to interact at many points in the development cycle – for example, users need to be available to provide information needed to carry out effective detailed interface design.

This interaction, or other external factors, often leads to changes being needed, which effectively vary the terms of the contract and so a careful change control procedure is needed. Each of these topics will now be tackled in a little more detail.

When a the contract is being negotiated, certain key points in the project can be identified where customer approval is needed before the project can proceed. For example, a project to develop a large system can be divided into increments. For each increment there could be an interface design phase, and the customer needs to approve the designs before the increment is built. There could also be a decision point between increments.

Chapter 4 discusses incremental delivery.

For each decision point, the deliverables to be presented by the suppliers, the decisions to be made by the customer and the outputs from the decision point all need to be defined. These decision points have added significance if payments to the supplier are based on them. Not only the supplier but also the customer has responsibilities with respect to these decision points – for example, the supplier should not be unnecessarily delayed while awaiting customer approval of some interim deliverable.

Where work is contracted out there will be a general concern about the quality of that work. The ISO 12207 standard envisages the possibility of there being agents, employed independently of the supplier or customer, who will carry out verification, validation and quality assurance. It also allows for joint reviews of project processes and products, the nature of which needs to be clearly agreed when the contract is negotiated, otherwise the supplier might claim unwarranted interference in their work.

As the system is developed a need to change certain of the requirements often emerges. As noted earlier, essentially, this is varying the terms of the contract. Oral evidence is not normally admissible to contradict, add to, or vary the terms of a written contract, so that agreed changes need to be documented properly. An effective change control procedure is therefore needed to record requests for changes, along with the supplier's agreement to them and any fees for the additional work.

It could happen that the supplier does not meet one or more of their legal obligations. This might be through no fault of theirs, if, for example, the customer has caused the delay by being tardy in giving the necessary approvals for intermediate products. If no action is taken when the default occurs, this can be taken to imply that the customer in fact condones the failure and this can lead to the loss of a right to legal redress. The customer should therefore protect their legal rights by officially notifying the supplier as soon as possible that the failure has been recognized. It will be recalled that under English law any claim for liquidated damages should be based on actual losses. From the point where the default occurs, the customer needs to keep an accurate record of the actual losses incurred as a result of the default including any consequential losses.

10.6 Acceptance

When the work has been completed, the customer needs to take action to carry out acceptance testing. The contract might put a time limit on how long acceptance testing can take, so the customer must be organized to carry out this testing before the time limit for requesting corrections expires.

We have already noted that some software houses are rather cursory with their pre-acceptance testing: the implication seeming to be that they would rather the users spent their time on testing than they themselves. This imposition can be reduced by asking to approve the supplier's internal test plans. An associated pitfall is that once the main development work is completed, the supplier not unnaturally wants to reallocate the most productive staff to other projects. The customer can find that all their problem reports are being dealt with by relative junior members of the supplier's staff, who might not be familiar with all aspects of the delivered system.

Part or all of the payment to the supplier will depend on this acceptance testing. Sometimes part of the final payment will be retained for a period of operational running and is eventually paid over if the levels of reliability are as contracted for. There is usually a period of warranty during which the supplier should fix any errors found for no charge. The supplier might suggest a very short warranty period of say 30 days. It is in the customer's interests to negotiate a more realistic period of say at least 120 days.

10.7 Summary

Some of the key points in this chapter have been:

- the successful contracting out of work requires considerable amounts of management time;
- it is easier to gain concessions from a supplier before a contract is signed than afterwards;
- alternative proposals need to be evaluated as far as possible by comparing costs over the whole lifetime of the system rather than just the acquisition costs;
- a contract will place obligations on the customer as well as the supplier;
- contract negotiation should include reaching agreement on the management of the supplier-customer relationship during the execution of the project.

10.8 Further exercises

1. At IOE, the management are considering 'out-sourcing' the maintenance accounting system, that is, getting an outside specialist organization to take over the operation, maintenance, and support activities associated with the

system. Write a short memorandum to management outlining the advantages and disadvantages of such a re-organization.

2. In each of the following cases discuss whether the type of application software to be adopted would be most likely to be bespoke, off-the-shelf or COTS.

   (a) A college requires a student fees application. It is suggested that the processes required in the application are similar to those of any billing system with some requirements that are peculiar to the administration of higher education.
   (b) A computer-based application is needed at IOE to hold personnel details of staff employed.
   (c) A national government requires a system that calculates, records and notifies individual tax-payers about income tax charges.
   (d) A hospital needs a knowledge-based system to diagnose the causes of eye complaints.

3. The schedule of charges per function point shown in Table 10.1 has higher rates for larger systems. Give arguments explaining why this might be justified and also arguments against.

4. Table 10.2 has a charge of 25% and 50% of the normal rate for deleting transactions from an application. This seems to be rather high for simply removing code. What work would be involved in deleting functionality that could justify this cost?

5. Assume that IOE has decided on a COTS solution that will replace the whole of the existing maintenance accounting system rather than simply plugging in additional modules to deal with groups accounts. Write a memorandum that Amanda could send to IOE's legal department outlining the important provisions that a contract to supply this system should have.

Chapter 11

Managing people and organizing teams

OBJECTIVES

When you have completed this chapter, you will be able to:

- identify some of the factors that influence people's behaviour in a project environment;
- select the most appropriate people for a project;
- understand the role of continuing training and learning;
- increase staff motivation;
- improve group working;
- use the most appropriate leadership styles;
- understand the characteristics of the various team structures that can be employed.

11.1 Introduction

We are going to examine some of the problems that Amanda and Brigette could meet when dealing with the staff who will be working for them. Where possible we will see if the findings of researchers can provide any ideas about what to do.

First, we will look at some aspects of organizational behaviour (OB) research. There will be three concerns: staff selection, staff development and, which will be dealt with in more detail, staff motivation.

We will look at how the project leader can encourage effective group working and decision making while balancing this, where needed, by purposeful leadership. The final part of the chapter looks at some of the more formal aspects of organizational structures.

The issues raised in this chapter have impacts at all stages of project planning and execution but in particular at the following points (see also Figure 11.1):

- although perhaps having little control over organizational structure, the project leader needs to be aware of its implications (Step 2);
- the scope and nature of activities can be set in a way that will enhance staff motivation (Step 4);
- many risks to project success relate to staffing (Step 6);
- the qualities of individual members of staff should be taken into account when allocating staff to activities (Step 7).

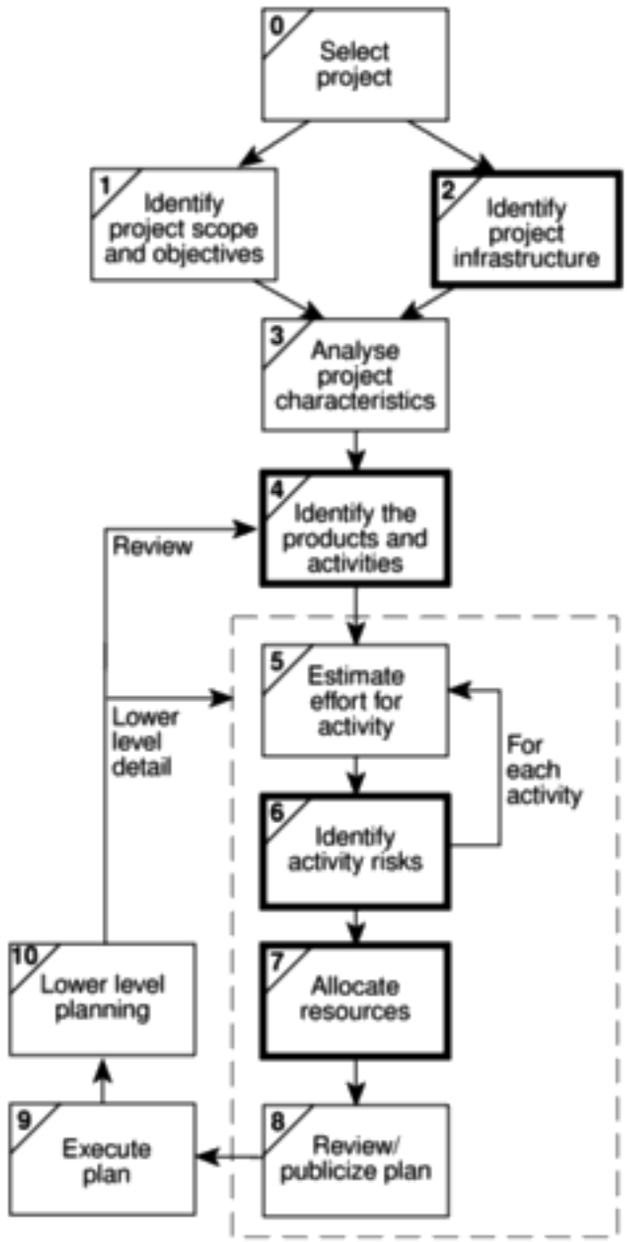


Figure 11.1 *Some places in the Step Wise framework where staffing concerns are important.*

11.2 Understanding behaviour

People with practical experience of working on projects invariably identify the handling of people as one of the most important aspects of project management. What people like Amanda and Brigette will want to know is whether the effective and sensitive management of staff comes only from experience or whether guidance can be usefully sought from writers on the topic.

The field of social science known as organizational behaviour (OB) helps. This has evolved theories that try to explain people's behaviour and that tend to be structured 'If A is the situation then B is likely to result'. Attempts are then made to observe behaviour or to conduct experiments where variables for A and B are measured and a statistical relationship between the two variables is sought. Unlike physical science, it is rarely, if ever, the case that it can be said that B must always follow A.

A major problem is that in the real world there is bound to be a very wide range of influences on a situation, many of which will not be apparent to the observer. It is therefore difficult to decide which set of research findings is relevant. A danger is that we end up with a set of maxims that are little better than superstitions. However, it is to hoped that by examining these questions people can become more sensitive and thoughtful about the problems involved.

In what follows, we will be making references to workers in the OB field such as Taylor, Mayo and McGregor. Rather than overwhelming the reader with references, we recommend the reader who is interested in exploring this topic further to look at Charles Handy's book. Where we have given references these tend to be for works related specifically to an IT environment.

Charles Handy, *Understanding organizations*, 4th edition, Penguin, 1993.

11.3 Organizational behaviour: a background

The roots of studies in OB can be traced back to work done in the late 19th and early 20th centuries by Frederick Taylor. By studying the way that manual workers did tasks, he attempted to work out the most productive way of doing these tasks. The workers were then trained to do the work in this way.

Taylor had three basic objectives:

- to select the best person for the job;
- to instruct such people in the best methods;
- to give incentives in the form of higher wages to the best workers.

Frederick Winslow Taylor, 1856–1915, is regarded as the father of 'scientific management' of which OB is a part.

'Taylorism' is often represented as crude and mechanistic these days. Interestingly though, the Taylorist approach is one that is adopted, in part, in modern sports coaching. A coach will attempt to get a javelin thrower, for example, to throw a javelin in a very exact manner in order to get the maximum effect. In the more mundane world of software development, the growth of

structured methods is an example of this emphasis on best practice. Both Amanda and Brigette will be concerned that tasks are carried out in the proper way. As we will see, more contentious is Taylor's emphasis on the exclusively financial basis of staff motivation, although Amanda and Brigette will be sure to find many colleagues who hold Taylor's view on the importance of 'performance-related pay'. Unfortunately, Amanda and Brigette are likely to have very little control over the financial rewards of their staff. However, they should be encouraged by findings that motivation rests not just on such rewards.

Elton Mayo and his colleagues did this research at the Hawthorne Works of Western Electric in Chicago, hence the 'Hawthorne Effect'.

During the 1920s, OB researchers discovered, while carrying out a now famous set of tests on the conditions under which staff worked best, that not only did a group of workers for whom conditions were improved increase their work-rates, but a control group, for whom conditions were unchanged, also increased their work-rates. Simply showing a concern for what workers did increased productivity. This illustrated how the state of mind of workers influenced their productivity.

The cash-oriented view of work of some managers can thus be contrasted with a more rounded vision of people in their place of work. The two attitudes were labelled *Theory X* and *Theory Y* by Donald McGregor.

Theory X holds that:

- the average human has an innate dislike of work;
- there is a need therefore for coercion, direction and control;
- people tend to avoid responsibility.

Theory Y, on the other hand, holds that:

A 'reward' does not have to be a financial reward – it could be something like a sense of achievement.

- work is as natural as rest or play;
- external control and coercion are not the only ways of bringing about effort directed towards the company's ends;
- commitment to objectives is a function of the rewards associated with their achievement;
- the average human can learn to accept and further seek responsibility;
- the capacity to exercise imagination and other creative qualities is widely distributed.

One way of judging whether a manager espouses Theory X or Theory Y is to observe how the manager's staff react when the boss is absent: if there is no discernible change, then this is a Theory Y environment; if everyone visibly relaxes, it is a Theory X environment. McGregor's distinction between the two theories also draws attention to the way that expectations influence behaviour. If a manager (or teacher) assumes that you are going to work diligently and do well, then you are likely to try and meet these expectations.

11.4 Selecting the right person for the job

Taylor stressed the need for the right person for the job. Many factors, such as the use of software tools and methodologies, affect programming productivity. However, one of the biggest differences in software development performance is among individuals. As early as 1968, a comparison of experienced professional programmers working on the same programming task found a ratio, in one case, of 1:25 between the shortest and longest time to code the program and, more significantly perhaps, of 1:28 for the time taken to debug it. Amanda and Brigette should therefore be rightly concerned to get the best possible people working for them.

B. W. Boehm considered the quality of staff the most important influence on productivity when constructing the COCOMO software cost models (Chapter 5).

What sort of characteristics should they be looking for? Should they go, for example, for the experienced programmer or the new graduate with the first class mathematics degree? It is extremely dangerous to generalize but looking specifically at behavioural characteristics, the American researcher Cheney found that the most important influence on programmer productivity seemed to be experience. Mathematical aptitude had quite a weak influence in comparison.

P. M. Cheney 'Effects of Individual Characteristics, Organizational Factors and Task Characteristics on Computer Programmer Productivity and Job Satisfaction' in *Information and Management*, 7 (1984).

Amanda and Brigette will want staff who can communicate well with each other and, more importantly, with users. They well have some difficulties here. The American researchers Couger and Zawacki found that computing people would appear to have much weaker 'social needs' than people in other professions. They quote Gerald Weinberg: '*If asked, most programmers probably say they prefer to work alone where they wouldn't be disturbed by other people.*' This is reflected in the problem that people attracted to writing software, and are good at it, will not make good managers later in their careers.

J. D. Couger and R. A. Zawacki 'What motivates DP Professionals?' in *Datamation*, 24 (1978).

The recruitment process

Although this is an important matter, it has to be stressed that often project leaders have little choice about the people who will make up the teams – they have to make do with the 'materials that are to hand'. Recruitment might very well be regarded as an organizational responsibility: you might be recruiting someone who will, over a period of time, work in many different parts of the same organization.

Meredith Belbin usefully distinguishes between *eligible* and *suitable* candidates. An eligible candidate is one whose CV (curriculum vitae or résumé) shows, for example, the 'right' number of years in some previous post and the 'right' paper qualifications. Suitable candidates are those who can actually do the job well. An easy mistake is to select an eligible candidate who is not in fact suitable. Suitable candidates who are not technically eligible can, on the other hand, be ideal candidates because, once in post, they are more likely to remain loyal to the organization. Belbin suggests that selection methods that centre on the assessment of actual skills rather than past experience and also a willingness to provide training to make good minor gaps in expertise can be a more effective way of placing suitable staff. It also seems to us to show that policies that avoid discrimination on the grounds of race, gender, age or irrelevant disabilities can be not just socially responsible but also a shrewd recruitment policy.

R. Meredith Belbin, *Team Roles At Work*, Butterworth-Heinemann, 1993

A general approach might be the following.

- **Create a job specification** Advice is needed, as there will be legal implications in an official document. However, formally or informally, the requirements of the job should be documented and agreed.
- **Create a job holder profile** Using the job specification, a profile of the person needed to carry out the job is constructed. The qualities, qualifications, education and experience required would be listed.
- **Obtain applicants** Typically, an advertisement would be placed, either within the organization or outside in the trade or local press. The job holder profile would be examined carefully to identify the medium most likely to reach the largest number of potential applicants at least cost. For example, if a specialist is needed it would make sense to advertise in the relevant specialist journal. The other principle is to give enough information in the advertisement to allow an element of self-elimination. By giving the salary, location, job scope and any essential qualifications, the applicants will be limited to the more realistic candidates.
- **Examine CVs** These should be read carefully and compared to the job holder profile – nothing is more annoying for all concerned than when people have CVs which clearly indicate that they are not eligible for the job and yet they are called for interview.
- **Interviews etc.** A number of different selection techniques can be tried, including aptitude tests, personality tests, and the examination of samples of previous work. All these methods must be related to specific qualities detailed in the job holder profile. Interviews are the most commonly used method. It is better if there is more than one interview session with an applicant and with each session there should not be more than two interviewers because a greater number reduces the possibility of follow-up questions and discussion. Some formal scoring system for the qualities being judged should be devised and interviewers should then decide scores individually which are then compared. An interview should be of a technical nature where the practical expertise of the candidate is assessed, or of a more general nature if not. In the latter case, a major part of the interview will in fact be evaluating and confirming what was stated in the CV – for example any time gaps in the education and employment history would be investigated, and the precise nature of jobs previously done would need to be explored.
- **Other procedures** References will need to be taken up where necessary, and a medical examination might be needed.

Exercise 11.1

A new analyst/programmer is to be recruited to work in Amanda's team at IOE. The intention is to recruit someone who already has some experience. Make a list

of the types of activities that the analyst/programmer should be capable of carrying out that can be used as the basis for a job specification.

11.5 Instruction in the best methods

This is the second concern that we have taken from Taylor. Obviously, there is a difference between loading pig iron (one of Taylor's studies) and writing C programs, but the principle of having established methods and procedures is, we hope, as well understood in software development as in steel-making.

When a new member of the team is recruited, the team leader will need to plan that person's induction into the team very carefully. Where a project is already well under way, this might not be easy. However, the effort should be made – it should pay off eventually as the new recruit will become a fully effective member of the team more quickly.

The need to take into account the time needed to acclimatize new staff was stressed in Chapter 8 on resource allocation.

The team leader should also be aware of the need to assess continually the training needs of their team members. Just as you formulate a user requirement before considering a new system, and you construct a job holder profile before recruiting a member of staff, so a training needs profile is drawn up for each staff member before you consider specific courses. Some training can be provided by commercial training companies. Where money is tight, other sources of training should be considered but training should not be abandoned altogether even if it consists only of a team member's being told to find out about a new software tool and then demonstrating it to colleagues. Of course the nice thing about external courses is that one gets to talk to colleagues from other organizations – but attending meetings of your local branch of one of the IS/IT professional associations can serve the same purpose.

The methods learnt need, of course, to be actually applied. Reviews and inspections should help to ensure this.

11.6 Motivation

The third concern that we noted from Taylor was that of motivating people to work. We are now going to look at some different models of motivation that have been proposed.

The Taylorist model

Taylor's viewpoint is reflected in the use of piece-rates in manufacturing industries and sales bonuses amongst sales forces. A problem that project leaders must be aware of is that piece-rates often cause difficulties if a new system is going to change work practices. If new technology is going to improve productivity, the question of adjusting piece-rates downwards to reflect this will be a sensitive issue.

Piece-rates are where workers are paid a fixed sum for each item they produce. Day-rates refer to payment for time worked.

Usually, radical changes in work practices have to be preceded by a move from piece-rates to day-rates.

Group norms are discussed further under group decision making.

Even where work practices are stable and output can be easily related to reward, people paid by the amount they produce will not automatically maximize their output in order to maximize their income. The amount of output will often by constrained by 'group norms', informal, even unspoken, agreements among colleagues about the amount to be produced.

Rewards have to be related in a simple and direct way to the work produced. Where a computer system is being produced, this is not easy. It is difficult to isolate and quantify work done, especially as system development and support is very much a team effort. Typical is the sentiment expressed by one member of staff in a study of software support work practices:

Quoted by Wanda J. Orlikowski in 'Evolving with Notes: Organizational change around groupware technology' in *Groupware & Teamwork*, edited by Claudio U. Ciborra, Wiley and Sons, 1996.

'*This support department does well because we're a team, not because we're all individuals. I think it's the only way the support team can work successfully.*'

In this kind of environment, a reward system that makes excessive distinctions between co-workers can be damaging to morale and eventually to productivity.

Exercise 11.2

A software development department wants to improve productivity by encouraging the re-use of existing software components. It has been suggested that this could be encouraged through financial rewards. To what extent do you think this could be done?

Maslow's hierarchy of needs

Different people are motivated by different things. Clearly money is a very strong motivator when you are broke. However, as the basic need for cash is satisfied, other motivators are likely to emerge. Abraham Maslow, an American psychologist, suggested that there is a hierarchy of needs. As lower levels of needs are satisfied then gradually higher level needs emerge. If these are then satisfied then yet another level of need will emerge. Basic needs are for things like food and shelter. The highest level need, according to Maslow, is the need for 'self-actualization', the feeling that you are completely fulfilling your potential.

In practice, the project leader must realise that people are likely to be motivated by different things at different stages of their life. For example, salary increases, while always welcome, probably have less of an impact on the more mature employee who is already relatively well-paid, than on a new and lowly-paid trainee. Older team-members might place more value on qualities of the job such as being allowed relative autonomy when they do their work, which shows respect for their judgment and sense of responsibility.

Exercise 11.3

Newspapers often report on the vast sums of money that are paid to the top executives of many companies. Does this mean that these people are at a low level

in the Maslow hierarchy of motivation? Do they really need all this money to be motivated? What do you think that the significance of these salaries really is?

Herzberg's two-factor theory

Certain things about a job might make you dissatisfied. If the causes of this dissatisfaction are removed, this does not necessarily make the job more exciting. On the basis of research into job satisfaction that Herzberg and his associates carried out there seemed to be two sets of factors about a job that were of importance:

- **hygiene or maintenance factors**, which can make you dissatisfied if they are not right, for example, the level of pay or the working conditions;
- **motivators**, which make you feel that the job is worthwhile, like a sense of achievement or the nature of the work itself.

Brigette, at Brightmouth College, is in an environment where it is difficult to compete with the high level of maintenance factors that can be provided by a large organization like IOE, but the smaller organization with its closer contact with the users is often able to provide better motivators.

Exercise 11.4

Identify three incidents or times when you felt particularly pleased or happy about something to do with your work or study. Identify three occasions when you were particularly dissatisfied with your work or study. Compare your findings with those of your colleagues and try to identify any patterns.

The expectancy theory of motivation

Amanda and Brigette will need to be aware of how the day-to-day ups and downs of system development affect motivation. A model of motivation developed by Vroom and his colleagues illustrates this. It identifies three influences on motivation:

- **expectancy**, the belief that working harder will lead to a better performance;
- **instrumentality**, the belief that better performance will be rewarded;
- **perceived value**, of the resulting reward.

Motivation will be high when all three factors are high. A zero level for any one of the factors can lead to a lack of motivation.

Imagine that you are trying to get a software package supplied by a third party to work. If you realize that you will never get it to work because of a bug in it, you

will give up. No matter how hard you work you will not be able to do any better (zero expectancy).

If you are working on a package for a user and, although you think you can get it to work, you discover that the user has started employing an alternative package and no longer needs this one, then you will probably feel you are wasting your time and give up (zero instrumentality).

Given that the users really do want the package, your reward in this set of circumstances might simply be a warm feeling that you have helped your colleagues and that they are grateful to you. If in fact, when the users employ the package all they do is complain and hold you responsible for any shortcomings, then you will probably avoid getting involved if they later ask for help implementing a different package (low perceived value of reward).

The Oldham–Hackman job characteristics model

Managers should try to group together the elements of the tasks that need to be carried out so that they form meaningful and satisfying assignments. Oldham and Hackman suggest that the satisfaction that a job gives is based on five factors. The first three factors make the job 'meaningful' to the person who is doing it:

- **skill variety**, the number of different skills that the job holder has the opportunity to exercise;
- **task identity**, the degree to which your work and its results are identifiable as belonging to you;
- **task significance**, the degree to which your job has an influence on others.

The other two factors are:

- **autonomy**, the discretion you have about the way that you do the job;
- **feedback**, the information you get back about the results of your work.

Couger and Zawacki found that programmers in general rated their jobs lower on these factors than other professions, while systems analysts and analyst-programmers rated them higher. Computer development people experienced about the same level of meaningfulness in their work as other, non-IT, professionals, but had lower perceptions of the degree of responsibility and knowledge of results of their work.

Cheney found that in the programming environment, the degree to which programmers got feedback on their work and the degree to which they could contribute to decision making had positive influences on both productivity and job satisfaction, although 'consideration', which was 'the degree to which the leader develops a work climate of psychological support, mutual trust and respect, helpfulness and friendliness', rated as less important.

In practical terms, activities should be designed so that, where possible, staff follow the progress of a particular product and feel personally associated with it.

Methods of improving motivation

- **Setting specific goals** These goals need to be demanding and yet acceptable to staff. Involving staff in the setting of goals helps to gain acceptance for them.
- **Providing feedback** Not only do goals have to be set but staff have to have regular feedback about how they are progressing.
- **Job design** Jobs can be altered to make them more interesting and give staff more feeling of responsibility.

Two measures are often used to enhance job design – job enlargement and job enrichment.

Job enlargement and job enrichment are based on the work of F. Herzberg.

- **Job enlargement** The scope of the job is increased so that the member of staff carries out a wider range of activities. It is the opposite of increasing specialization. For example, a programmer in a maintenance group might be given responsibility for specifying minor amendments as well as carrying out the actual code changes. It is significant that Couger and Zawacki found that programmer/analysts had a higher degree of job satisfaction than programmers.
- **Job enrichment** In this case, the job is changed so that the holder carries out tasks that are normally done at a higher, managerial, level. Staff might be given responsibility for ordering consumables, for scheduling their work or for quality control. With a programmer in a maintenance team, they might be given authority to accept requests for changes which involved less than five days' work without the need for their manager's approval.

11.7 Working in groups

Having discussed people as individuals, we move on to their place in groups. A key problem with major software projects is that they always involve working in groups, but as we have seen many people attracted to computer development find this difficult.

Formal groups can be subdivided into *command groups*, which are the departmental groupings that are seen on organization hierarchy diagrams and which reflect the formal management structure and *task groups* set up to deal with specific tasks. These call on people from different command groups and would typically be disbanded once the task has been completed.

11.8 Becoming a team

Simply throwing people together does not mean that they will immediately be able to work together as a team. Group feelings develop over a period of time. One suggestion is that teams go through five basic stages of development:

This classification is associated with B. W. Tuckman and M. A. Jensen.

- **forming** – the members of the group get to know each other and try to set up some ground rules about behaviour;
- **storming** – conflicts arise as various members of the group try to exert leadership and the group's methods of operation are being established;
- **norming** – conflicts are largely settled and a feeling of group identity emerges;
- **performing** – the emphasis is now on the tasks at hand;
- **adjourning** – the group disbands.

Where people are being put together into a team for the first time, then some specific team-building exercises can be undertaken. Some organizations, for example, send their management teams off on outward bound courses. Without going to these lengths, Amanda and Brigette might try and think of some training activity which could assist in team building.

R. Meredith Belbin *Management Teams: Why They Succeed or Fail*, Heinemann, 1981, contains a self-assessment questionnaire that can help identify to which role a person is best suited.

Valuable research has gone into looking at the best mix of personalities in a project team. Belbin studied teams working together on management games using various mixes of people. He initially tried putting all the people who were most able into one group. Surprisingly, these élite teams tended to do very badly – they argued a lot and as a result important tasks were often neglected.

Belbin came to the conclusion that teams needed a balance of different types of people.

- **The chair** Not necessarily a brilliant leader but must be good at running meetings, being calm, strong but tolerant.
- **The plant** Someone who is essentially very good at generating ideas and potential solutions to problems.

In *Team roles at work*, 1993, Belbin suggests that 'co-ordinator' and 'implementer' are better descriptions than 'chair' and 'team worker'. A new role is added: the 'specialist', the 'techie' who likes to acquire knowledge for its own sake.

- **The monitor-evaluator** Good at evaluating ideas and potential solutions and helping to select the best one.
- **The shaper** Rather a worrier, who helps to direct the team's attention to the important issues.
- **The team worker** Skilled at creating a good working environment, for example by 'jollying people along'.
- **The resource investigator** Adept at finding resources in terms of both physical resources and information.
- **The completer-finisher** Good at completing tasks.
- **The company worker** A good team player who is willing to undertake less attractive tasks if they are needed for team success.

A person can have elements of more than one type. On the other hand, about 30% of the people examined by Belbin could not be classified at all! To be a good team member you must be able to:

- time your interventions, that is, not overwhelm the others in the team;
- be flexible;
- be restrained;
- keep the common goals of the team in mind all the time.

Group performance

Are groups more effective than individuals working alone? Given the preference of many people attracted to software development for working on their own, this is an important question. In many projects, judgements need to be made about which tasks are best carried out collectively and which are best delegated to individuals to do on their own. As one manager at IBM was quoted as saying: '*Some work yields better results if carried out as a team while some things are slowed down if the work is compartmentalized on an individual basis*'. Part of the answer lies in the type of task being undertaken.

The IBM manager was quoted by Angelo Failla in 'Technologies for Co-ordination in a Software Factory' in *Groupware & Teamwork* edited by C. U. Ciborra, Wiley & Sons, 1996.

One way of categorizing group tasks is into:

- additive tasks;
- compensatory tasks;
- disjunctive tasks;
- conjunctive tasks.

Additive tasks are where the efforts of each participant are added together to get the final result, as in a gang of people clearing snow. The people involved are interchangeable.

With *compensatory tasks* the judgements of individual group members are pooled so that errors by some group members are compensated for by the inputs from others. An example of this would be where individual members of a group are asked to provide estimates of the effort needed to produce a piece of software and the results are then averaged. In these circumstances, group work is generally more effective than the efforts of individuals.

Code reviews could be seen as an example of a compensatory task.

With *disjunctive tasks* there is only one correct answer. The effectiveness of the group depends on:

- someone coming up with the right answer;
- the others recognizing it as being correct.

With this type of task, the group can only be as good as its best member and no better.

Conjunctive tasks are where progress is governed by the rate of the slowest performer. Software production where different staff are responsible for different modules seems to be a prime example of this. The overall task is not completed until every participant's work is complete. In this case co-operative attitudes are

productive: the team members who are more advanced need to ensure the meeting of group objectives by assisting those who are behind.

The source of the quotation is the paper by Failla that is cited above.

With all types of collective task, but particularly with additive ones, there is a danger of *social loafing*, where some individuals do not make their proper contribution. This can certainly occur with student group activities, but is not unknown in 'real' work environments. As one software developer has commented: '*[The contribution made to others] is not always recognized. Nor is the lack of any contributions … nobody points out those who fail to make any contributions. Like when there's somebody with vital skills and you ask him for help, but he doesn't provide it*'.

Exercise 11.5

Social loafing is a problem that students often encounter when carrying out group assignments. What steps can participants in a group take to encourage team members to 'pull their weight' properly?

11.9 Decision making

Before we can look more closely at the effectiveness with which groups can make decisions we need to look in general terms at the decision-making process.

Many of the techniques in Chapter 3 are attempts to make decision making more structured.

Decisions can be categorized as being:

- **structured**, generally relatively simple, routine decisions where rules can be applied in a fairly straightforward way;
- **unstructured**, more complex and often requiring a degree of creativity.

Another way of categorizing decisions is by the amount of *risk* and *uncertainty* that is involved.

Many of the techniques in Chapter 3 on project selection are based on the rational-economic model.

Yet another distinction is between the rational-economic model and the satisficing model. The *rational-economic* model of decision making is the basis of classical economics. It predicts, for example, that a prospective buyer of personal computer equipment will purchase goods at the lowest possible price. This assumes that the decision maker has a complete knowledge of the state of the market. In order to achieve this, days, weeks, or months could be spent phoning dealers.

Some research has found that organizations with the most comprehensive solution-seeking techniques are often the poorer financial performers!

Sensible people probably follow a *satisficing* approach and would look at a limited number of representative outlets to get a general idea of prices. Any potential loss of money through having missed an even lower offer would probably be offset by the savings in time, phonecalls, travel and so on.

Some mental obstacles to good decision making

In this book we have rightly stressed a structured, rational, approach to decision making. Many management decisions in the real world, however, made under

pressure and based on incomplete information, are largely intuitive. We have to accept the role of intuition but must be aware that there are some mental obstacles to effective intuitive thinking, for example:

Faulty heuristics Heuristics mean rules of thumb. Rules of thumb can be useful but there are dangers:

- they are based only on the information that is to hand and this can be misleading;
- they are based on stereotypes, such as accepting a Welshman into a male voice choir without an audition because of the 'well-known fact' that the Welsh are a great singing nation.

Escalation of commitment This refers to the way that once you have made a decision it is increasingly difficult to alter it even in the face of evidence that it is wrong.

Information overload It is actually possible to be presented with too much information so that you 'cannot see the wood for the trees'.

Group decision making

There will be occasions where Amanda at IOE, for instance, will want to consult her whole project team about some problem. With a project team, different specialists and points of view can be brought together. Decisions made by the team as a whole are more likely to be accepted than those that are imposed upon it.

A different type of participatory decision-making might occur when end users are consulted about the way a projected computer system is to operate.

Assuming that the meetings are genuinely collectively responsible and have been properly briefed, research shows that groups are better at solving complex problems where the members of the group have complementary skills and expertise. The meeting allows them to communicate freely and to get ideas accepted.

Groups are less effective when dealing with poorly structured problems, which need creative solutions. Brainstorming techniques have been developed to help groups in this situation but research shows that people often come up with more ideas individually than in a group. Where the aim is to get the involvement of end users of a computer system, then prototyping and participatory approaches such as Joint Application Development (JAD) might be adopted.

JAD has been already discussed in Chapter 4.

Obstacles to good group decision making

Amanda finds that group decision making has some disadvantages: it is time consuming; it can in some cases stir up conflicts within the group; and decisions can be unduly influenced by dominant members of the group.

Conflict could, in fact, be less than might be expected. Experiments have shown that people will modify their personal judgements to conform to *group norms*. These are common attitudes that are developed by a group over a period of time.

Once established group norms can survive many changes of membership in the group.

You might think that this would tend to moderate the more extreme views that some individuals in the group might hold. In fact, people in groups often make

decisions that carry more risk than where they have to make the decision on their own. This is known as the *risky shift*.

Measures to reduce the disadvantages of group decision making

One method of making group decision making more efficient and effective is by training members to follow a set procedure. The *Delphi technique* endeavours to collate the judgements of a number of experts without actually bringing them face-to-face. Given a problem, the following procedure is carried out:

- the co-operation of a number of experts is enlisted;
- the problem is presented to the experts;
- the experts record their recommendations;
- these recommendations are collated and reproduced;
- the collected responses are recirculated;
- the experts comment on the ideas of others and modify their recommendations if so moved;
- if the leader detects a consensus then the process is stopped, otherwise the comments are recirculated to the experts.

The big problem with this approach used to be that because the experts could be geographically dispersed the process was time consuming.

Exercise 11.6 What developments in information technology would be of particular assistance to use of the Delphi technique?

11.10 Leadership

When Amanda and Brigette first took on project management responsibilities, one of their private anxieties was a fear that they would not have enough personal authority – that staff would not take them seriously. Leadership is generally taken to mean the ability to influence others in a group to act in a particular way in order to achieve group goals. A leader is not necessarily a good manager or vice versa, because managers have other roles to play, such as those of organizing, planning and controlling.

Authorities on this subject have found it very difficult to agree a list of the common characteristics of good leaders. It would, however, seem safe to say that they seem to have a greater need for power and achievement and have more self-control and more self-confidence than others.

Leadership is based on the idea of some kind of authority or power, although leaders do not necessarily have much formal authority. This power comes from

either the person's position (*position* power) or from the person's individual qualities (*personal* power) or can be a mixture of the two. Position power has been further analysed into:

These ideas are associated with the work of J. R. P. French and B. H. Raven.

- **coercive power**, the ability to force someone to do something by threatening punishment;
- **connection power**, which is based on having access to those who have power;
- **legitimate power**, which is based on a person's title conferring a special status;
- **reward power**, where the holder can confer rewards on those who carry out tasks to their satisfaction.

Personal power, on the other hand, can be further analysed into:

- **expert power**, which comes from being the person who is able to do a specialized task;
- **information power**, where the holder has access to information that others do not;
- **referent power**, which is based on the personal attractiveness of the leader.

Exercise 11.7

What kinds of power (as defined above) would the following people have?

i. An internal auditor looking at the payroll system at Brightmouth College.

ii. A consultant who is called in to advise International Office Equipment about ways of improving software development productivity.

iii. The principal of Brightmouth College who has told staff that they must accept a new contract or face the sack.

iv. Brigette in respect to the users of the college payroll system.

v. Amanda in respect of the people in the project team developing the group maintenance accounts application.

Leadership styles

We have already suggested that Amanda and Brigette were initially concerned about establishing their personal authority. Balanced against this is the need to involve the staff in some of the decision making in order to make the best use of expertise and to gain commitment. Amanda and Brigette will need to judge when they must be authoritative and insist on things and when they must be more flexible and tolerant. Amanda, for example, might decide to be very democratic when formulating plans, but once the plans have been agreed, to insist on a very

disciplined execution of the plan. Brigette, on the other hand, might find at Brightmouth College that she alone has the technical expertise to make some decisions, but, once she has briefed people on what needs to be done, they expect to be left alone to get on with the job as they best see fit.

Attempts have been made to measure leadership styles on two axes: directive vs. permissive and autocratic vs. democratic:

This approach is associated with Rensis Likert.

- **directive autocrat** makes decisions alone with close supervision of their implementation;
- **permissive autocrat** makes decision alone but gives subordinates latitude in implementation;
- **directive democrat** makes decisions participatively but closely supervises their implementation;
- **permissive democrat** makes decisions participatively and gives subordinates latitude in implementation.

It should be emphasized that there is no one best style of management – it depends on the situation.

Another axis on which there have been attempts to measure management qualities has been on the degree to which a manager is *task-oriented*, that is, the extent to which the execution of the task at hand is paramount, and the degree to which the manager is concerned about the people involved (*people orientation*). It is perhaps not surprising that subordinates appear to perform best with managers who score highly in both respects.

Work environments vary according to the amount of control that can be exerted over the work. Some jobs are routine and predictable (as when dealing with batched computer output). Others may be driven by outside factors (as in the case of a help-desk) or are situations where future direction is uncertain (for example, at the early stages of a feasibility study). Where there is a high degree of uncertainty, subordinates will seek guidance from above and welcome a task-oriented management style. As uncertainty is reduced, the task-oriented manager is likely to relax and to become more people-oriented and this will have good results. People-oriented managers are better where staff can control the work they do and know what to do without referring matters to their line managers. It is then argued that if control becomes even easier the people-oriented manager will be tempted to get involved in more task-centred questions and that this may have undesirable results.

Research findings also show that where team members are relatively inexperienced a task-oriented approach is most effective. As group members mature, consideration for their personal needs and aspirations becomes more valued. Where maturity is very high, then there is no need for a strong emphasis on either of these approaches.

Exercise 11.8

What in your view would be the most appropriate management style when dealing with the following subordinates?

i. At Brightmouth College, a former member of the local authority who has dealt with the college payroll for several years and who has been employed by the college to set up and manage the new payroll section.

ii. At IOE, a new trainee analyst–programmer who has just joined Amanda's group.

iii. At IOE, a very experienced analyst–programmer aged 45, who was recruited into the software development department some time ago from the accounts department and who has been dealing with system support for the old maintenance accounts system that is now being revised.

11.11 Organizational structures

Formal versus informal structures

While organizational structures can have an enormous impact on the way a project is conducted, it is something that project leaders such as Amanda at IOE can often do little to change.

The *formal* structure is the one that is expressed in the staff hierarchy chart. It is basically concerned with *authority*, about who has which boss. It is backed by an *informal* structure of contacts and communication that grows up spontaneously among members of staff during the course of work. When the unexpected happens it is often this system that comes into play. Over a period of time, the advantages and disadvantages of different organizational structures tend to even out – the informal organization gets built up and staff find unofficial ways of getting around the obstacles posed by the formal structure.

Hierarchical approach

The 'traditional' management structure is based on the concept of the *hierarchy* – each member of staff has only one manager, while a manager will have responsibility for several members of staff. Authority flows from the top down through the structure. A traditional concern has been with the *span of control* – the number of people that a manager can effectively control.

Staff versus line

Staff in organizations can often be divided into *line* workers who actually produce the end product and support *staff* who carry out supporting roles. In some organizations that produce software for the market or as a component of a larger product which is sold, the software specialists might be seen as part of the line. In a financial organization, on the other hand, the information systems department would probably be seen as part of the support staff.

Departmentalization

In drawing up a structure, the question of *differentiation* crops up. This is the question of how the organization is to be departmentalized. This is often based on staff specialisms, product lines, categories of customer or geographical location, for example.

In the case of software development, it is usually the case that either a *functional* or a *task-oriented* approach is used. With functional departmentalization, systems analysts might be put in a group separate from the programmers. The programmers would act as a pool from which resources may be drawn for particular tasks. With a task-oriented approach, the programmers and systems analysts are grouped together in one project team. The project team might be gathered in order to implement a specific long-term project or might exist on a permanent basis to service the needs of a particular set of users.

One advantage of the functional approach is that it can lead to a more effective use of staff. Programmers can be allocated to jobs as needed and be released for other work when a particular task is completed. For instance, in a project team there are bound to be periods of greater and lesser coding activity and programmers might find there are spells when they are under-utilized. The functional organization will also make it easier for programmers to have careers that are technically oriented – there will probably be a career structure within the software development department that allows the programmer to rise without having to change specialism. This type of organization should also encourage the interchange of new technical ideas among technical staff and the promulgation of company wide standards.

A disadvantage is that having two separate departments can lead to communication problems, especially if a programmer is unfamiliar with the application area. There will also be problems with software maintenance – here it is helpful to have programmers who have built up a familiarity with particular parts of the application software. Users might prefer the established project team approach because, when they require new software features, they will already have a group dedicated to their needs and will not find themselves in the position of always having to fight other departments for development resources. The project team structure tends to favour a pattern of career progression where programmers eventually become systems analysts.

A third method of departmentalization is based on lifecycle phase. Here there are separate teams for development and maintenance. Some staff can concentrate in a focused and sustained manner on developing new applications with few interruptions, while other teams, more oriented towards service and support, deal with maintenance.

Some organizations have attempted to get the best of all worlds by having a *matrix* structure. In this case the programmer would have two managers: a project leader who would give day-to-day direction about the work in hand and a programming manager who would be concerned about such things as career development.

Centralized versus decentralized group structures

At the level of a project group, a decentralized organization would mean that the group members would tend to make major decisions collectively and that there would be a large degree of free communication among group members. With the centralized approach the group would be broken down into sections, each of which would be directed by a leader who communicates on behalf of the section with other groups.

Decentralized groups, because of the time taken to debate things, tend to work more slowly. They are likely to be affected by the establishment of group norms and the influence of the risky shift, which has already been described. However, they are better at dealing with complex problems while the centralized group organization deals more effectively with simple problems.

The discussion of centralized versus decentralized groups assumes that software development work has to be done as a group. In fact, given the preference of many software developers for working on their own, an organization where each programmer works in isolation can be envisaged – indeed there are software houses that are based on people working at home.

Egoless programming

In the early days of computer development, managers tended to think of the programmer as communing mysteriously with the machine. The tendency was for programmers to see programs as being an extension of themselves and to feel over-protective towards them. The effects of this on the maintainability of programs can be imagined. Gerald Weinberg made the then revolutionary suggestion that programmers and programming team leaders should read other people's programs. Programs would become in effect the common property of the programming group and programming would become 'egoless'. Peer code reviews are based on this idea. Weinberg's ideal programming team was a decentralized group freely communicating within itself.

G. M. Weinberg, *The Psychology of Computer Programming*, Van Nostrand Reibold, 1971.

Chief programmer teams

The larger the decentralized group, the slower it will get, because of the increased communication. On really large time-critical projects, a more formalized centralized structure is essential. Brooks pointed out the need for design consistency when producing a large complex system and how this might be difficult when there are a large number of people involved in producing a piece of software. One suggestion was to try to reduce the number of people actually creating software but to make these programmers as productive as possible by giving them as much support as possible.

The result of this train of thought was the *chief programmer* team. The chief programmer is the person who defines the specification, and designs, codes, tests and documents the software. There is also a *copilot*, with whom the chief programmer can discuss problems and who writes some code. They are supported by an *editor* to write up the documentation drafted by the chief programmer, a

Brooks' *Mythical Man-Month* has already been referred to. He was in charge of the huge team that created the operating system for the IBM 360 range.

program clerk to maintain the actual code, and a *tester*. The general idea is that this team is under the control of a single unifying intellect.

The chief programmer concept was used on the influential *New York Times* data bank project, where many aspects of structured programming were tried out. In this case, each chief programmer managed a senior level programmer and a program librarian. Additional members could be added to the team on a temporary basis to deal with particular problems or tasks.

The problem with this kind of organization is getting hold of really outstanding programmers to carry out the chief programmer role. There are also the dangers of information overload on the chief programmer, and of staff dissatisfaction among those who are there simply to minister to the needs of the superstar chief programmers.

Controlled decentralized groups

This compromise structure has been suggested and seems to follow common industry practice. A project team is made of groups under the leadership of senior programmers. Within these groups there is free communication and a practice of reviewing each others' work. Communication with other groups is at senior programmer level, while a project leader has overall authority.

11.12 Conclusion

Some of the important points that have been made in this chapter are:

- people may be motivated by money, but they are motivated by other things as well;
- both staff selection and the identification of training needs should be done in an orderly, structured, way where requirements are clearly defined first;
- thoughtful job design can increase staff motivation;
- consideration should be given, when forming a new project team, to getting the right mix of people and to planning activities that will promote team building;
- group working is more effective with some types of activity than others;
- different styles of leadership are needed in different situations;
- the people who need to communicate most with each other should be grouped together organizationally.

11.13 Further exercises

1. An organization has detected low job satisfaction in the following departments:

- the system testing group;

- the computer applications help desk;
- computer batch input.

How could these jobs be redesigned to give more job satisfaction?

2. In Exercise 11.1, a job specification was requested.
   (a) Write a job holder profile of the sort of person who would be able to fulfil the specification in terms of qualities, qualifications, previous education and experience.
   (b) For each element in the job holder profile that you have produced in (a) above, describe ways of finding out whether an applicant has met the requirement.
3. To what extent is the Belbin approach to balanced teams compatible with having chief programmer teams?
4. If you have been involved recently in a group activity or project, try and categorize each participant according to the Belbin classification. Were there any duplications or gaps in any of the roles? Did this seem to have any impact on progress?
5. Three different mental obstacles to good decision making were identified in the text: faulty heuristics, escalation of commitment and information overload. What steps do you think can be taken to reduce the danger of each of these?
6. In Exercise 11.8, the management style most appropriate for each of three different situations was asked for. Go back and consider how you as a manager would respond to each of these three situations in terms of practical things to do or avoid.