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;
- appreciate the need for careful planning, monitoring and control;
- identify the stakeholders of a project and their objectives and ways of defining the success in meeting those objectives.

1.1 Introduction

What is 'software project management'? What makes it different from management in general? Is *software* project management different from other types of project management? To answer these questions we need to look at some key ideas about the planning, monitoring and control of software projects. We will see that projects are largely about meeting objectives. Projects to produce software are only worthwhile if they satisfy real needs and so we will examine how we can identify the stakeholders in a project and their objectives. Having identified those objectives, ensuring 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: e.g. a public works scheme', *Longman Dictionary of the English Language*, 1991.

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

The definition of a project as being planned assumes that to a large extent we can determine how we are going to carry out a task before we start. There may be some projects of an exploratory nature where this might be quite difficult. Planning is in essence thinking carefully about something before you do it – and even in the case of uncertain projects this is worth doing as long as it is accepted that the resulting plans will have provisional and speculative elements. Other activities, relating, for example, to routine maintenance, might have been performed so many times that everyone involved knows exactly what needs to be done. In these cases, planning hardly seems necessary, although procedures might need to be documented to ensure consistency and to help newcomers to the job.

The types of activity that will benefit most from conventional project management are likely to lie between these two extremes – see Figure 1.1.

Programme management is often used to coordinate activities on concurrent jobs – see Appendix C.

There is a hazy boundary between the non-routine project and the routine job. The first time you do a routine task it will be like a project. On the other hand, a project to develop a system similar to previous ones that you have developed will have a large element of the routine.

The characteristics which distinguish projects can be summarized 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 pre-determined time span;
- 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.

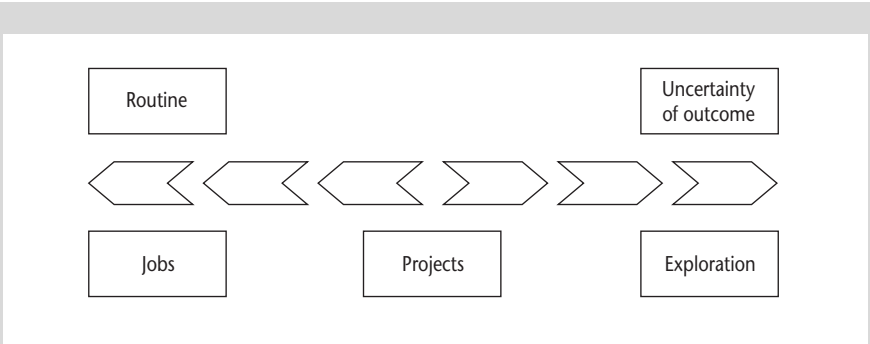

Figure 1.1 Activities most likely to benefit from project management

The more any of these factors apply to a task, the more difficult that task will be. Project size is particularly important. A project that employs 200 project personnel is going to be trickier to manage than one with 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 a common European currency
- 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 second year programming assignment for a computing student
- writing an operating system for a new computer
- installing a new version of a word processing package 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 which 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

Brooks, F. P. (1987). 'No silver bullet: essence and accidents of software engineering'. This essay has been included in *The Mythical Man-Month*, Anniversary Edition, Addision-Wesley, 1995.

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 which make them different.

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.

Conformity The 'traditional' engineer is usually working with physical systems and physical materials like cement and steel. These physical systems can have some complexity, but are governed by physical laws that are consistent. Software developers have to conform to the requirements of human clients. It is not just that individuals can be inconsistent. Organizations, because of lapses in collective

memory, in internal communication or in effective decision-making can exhibit remarkable 'organizational stupidity' that developers have to cater for.

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 Contract management and technical project management

Many organizations contract out IT development to outside specialist developers. In such cases, the client organization will often appoint a 'project manager' to supervise the contract. This project manager will be able to delegate many technically oriented decisions to the contractors. For instance, the project manager will not be concerned about estimating the effort needed to write individual software components as long as the overall project is fulfilled within budget and on time. On the supplier side, there will need to be project managers who are concerned with the more technical management issues. This book leans towards the concerns of these 'technical' project managers.

1.5 Activities covered by software project management

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

A software project is not only concerned with the actual writing of software. In fact, where a software application is bought in 'off the shelf', there may 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.

Usually there are three successive processes that bring a new system into being – see Figure 1.2.

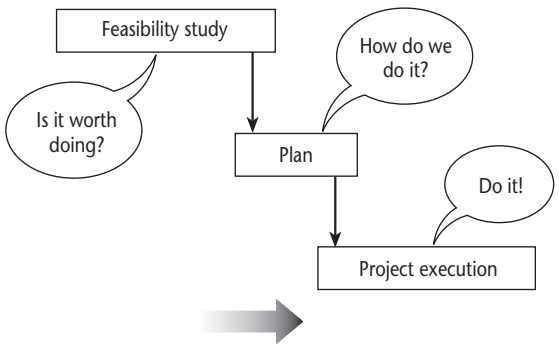

Figure 1.2 The feasibility study/plan/execution cycle

An outline of the content of a feasibility study is shown in Annex 1 to this chapter.

Appendix C contains a brief description of programme management.

The PRINCE2 method which is described in Appendix A takes this planning by stages approach.

Annex 2 to this chapter, has an outline of the content of a plan.

1. **The feasibility study** This is an investigation into whether a prospective project is worth starting. Information is gathered about the requirements of the proposed application. 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 – and have its own planning sub-phase. The study could be part of a strategic planning exercise examining and prioritizing a range of potential software developments. Sometimes an organization has a policy where a group of projects is planned as a *programme* of development.
2. **Planning** If the feasibility study produces results which indicate that the prospective project appears viable, planning of the project can take place. However, 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. The execution of a project often contains *design* and *implementation* sub-phases. Students new to project planning often find it difficult to separate planning and design, and often the boundary between the two can be hazy. Essentially, design is thinking and making decisions about the precise form of the *products* that the project is to create. In the case of software, this could relate to the external appearance of the software, that is, the user interface, or the internal architecture. The plan lays down the *activities* that have to be carried out in order to create these products. Planning and design can be confused because at the most detailed level, planning decisions are influenced by design decisions. For example, if a software product is to have five major components, then it is likely that there will be five groups of activities that will create them.

Figure 1.3 suggests that these stages must be done strictly in sequence – we will see in Chapter 4 that other, iterative, approaches can be adopted. However, the actual activities listed here would still be done.

Individual projects are likely to differ considerably but a classic project life cycle is shown in Figure 1.3. The stages in the life cycle 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

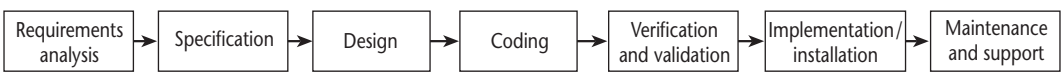

Figure 1.3 A typical project life cycle

small system which satisfies some, but not all, of the users' needs at a low price might be compared to a system with more functions but a higher price.

- *Specification* Detailed documentation of what the proposed system is to do.
- *Design* A design has to be drawn up which meets the specification. As noted earlier, this design will be in two stages. One will be the external or user design concerned with the external appearance of the application. The other produces the physical design which tackles the way that the data and software procedures are to be structured internally.
- *Coding* This may refer to writing code in a procedural language such as C or Ada, or could refer to the use of an application-builder such as Microsoft Access. Even where software is not being built from scratch, some modification to the base package could 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 latter case it includes setting up operational 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 is 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 a 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 package 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 package is to be used, how would this project differ from one where the software was to be written from scratch?

1.6 Plans, methods and methodologies

A plan for an activity must be based on some idea of a *method* of work. To take a simple example, if you were asked to test some software, even though you do not know anything about the software to be tested, you could assume that you would need to:

- analyse the requirements for the software;
- devise and write test cases that will check that each requirement has been satisfied;

- create test scripts and expected results for each test case;
- compare the actual results and the expected results and identify discrepancies.

While a *method* relates to a type of activity in general, a *plan* takes that method (and perhaps others) and converts it to real activities, identifying for each activity:

- its start and end dates;
- who will carry it out;
- what tools and materials will be used.

'Material' in this context could be information, for example, a requirements document.

With complex procedures, several methods may be deployed, in sequence or in parallel. The output from one method might be the input to another. Groups of methods or techniques are often referred to as *methodologies*. Object oriented design, for example, can be seen as a methodology made up of several component methods.

1.7 Some ways of categorizing software projects

See Simon Bennett, Steve McRobb and Ray Farmer, *Object Oriented Systems Analysis and Design using UML*, McGraw-Hill, 1999.

It is important to distinguish between the main types of software project because what is appropriate in one context may not be so in another. For example, some have suggested that the object-oriented approach, while useful in many contexts, might not be ideal for designing applications to be built around relational databases.

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

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

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.

On the other hand, the project may be required to meet certain objectives. There could be several ways of achieving these objectives. A new information system might be implemented to improve some service to users inside or outside an organization. The level of service that is the target would be the subject of an agreement 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 system 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?

1.8 What is management?

The Open University Software Project Management module (1987) suggested 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, etc.;
- 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 1.5 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 manager 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 programming 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 may 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.

1.9 Problems with software projects

One way of deciding what ought to be covered in 'software project management' is to consider what the problems are that it should address.

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 have become dispersed throughout the organization: there are management systems rather than managers.

Nevertheless the successful project will normally have one person who is charged with its success. Such people are focused on the overcoming of obstacles to success – they are primarily trouble-shooters and their job is likely to be shaped by the problems that confront the project. A survey of managers published by Thayer, Pyster and Wood identified the following commonly experienced problems:

Further details of the survey can be found in 'Major issues in software engineering project management', *IEEE Transactions on Software Engineering*, 7, 333–342, 1981.

- 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.

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 Computing and Information Systems course 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.

Many of the problems that were identified by the students stem from poor communications. A factor that contributes to this is the wide range of IT specialisms – an organization may be made up of lots of individuals each of whom is expert in one set of software techniques and tools but ignorant of those used by his or her colleagues. Communication problems are therefore bound to arise.

Stephen Flower's *Software Failure, Management Failure*, Wiley, 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 US 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 UK 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 clerically three months of records – 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 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 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!

1.10 Setting 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 may be several sets of users of a system and there may be several different groups of specialists involved in 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 which has overall authority over what the project is to achieve.

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

This authority is often held by a *project steering committee* (or *project board*) 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.

Sub-objectives and goals

Defining sub-objectives requires assumptions about how the main objective is to be achieved.

Setting objectives can be used to guide and motivate individuals and groups of staff. To be useful, though, the objective set for an individual must be something that is within the control of that individual. An objective might be set that the software application to be produced must pay for itself by reducing staff costs over two years. As an overall business objective this might be reasonable. For software developers it would be unreasonable as, though they can control development costs, any reduction in operational staff costs depends not just on them, but also on the operational management after the application has 'gone live'. What would be appropriate would be to set a *goal* or sub-objective for the software developers to keep development costs within a certain budget.

Thus, objectives will need to be broken down into goals or sub-objectives. Here we say that in order to achieve the objective we must achieve certain goals first. These goals are 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 1.6 Bearing in mind the above discussion of objectives, comment on the appropriateness of the wording of each of the following 'objectives' for software developers:

(i) to implement the new application on time and within budget;

(ii) to implement the new software application with as few software errors as possible that might lead to operational failures;

(iii) to design a system that is user friendly;

(iv) to produce full documentation for the new system.

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, e.g. 'Did we install the new software by 1st June?'

One of the authors, Bob Hughes, explores measurement issues in detail in the companion textbook *Practical Software Measurement*, McGraw-Hill, 2000.

A measure of effectiveness will usually be related to the installed operational system. 'Mean time between failures' (mtbf) might, for example, be used to measure reliability. Such measures are *performance* measures and, as such, can only be taken once the system is operational. Project managers will want to get some idea of the likely performance of the completed system as it is being constructed. They will therefore be seeking *predictive* measures. For example. a large number of errors found during code inspections might indicate potential problems later with reliability.

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?

1.11 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 data base 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 may 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.

Different types of stakeholder may 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.)

B. W. Boehm and R. Ross 'Theory W Software Project Management: Principles and Examples', in B. W. Boehm (ed.) *Software Risk Management*, CS Press, Los Alamitos, CA, 1989.

Exercise 1.8 Identify the stakeholders in the Brightmouth College payroll project.

1.12 The business case

Most projects need to have a justification or business case: the effort and expense of pushing the project through must be seen to be worthwhile in terms of the benefits that will eventually be felt. A cost–benefit analysis will often be part of the project's feasibility study. This will itemize and quantify the project's costs and benefits. The benefits will be affected by the completion date: the sooner the project is completed, the sooner the benefits can be experienced. The quantification of benefits will often require the formulation of a *business model* which explains how the new application can generate the claimed benefits.

A simple example of a business model is that a new web-based application might allow customers from all over the world to order a firm's products via the internet, increasing sales and thus increasing revenue and profits.

Any project plan must ensure that the business case is kept intact, for example:

- that development costs are not allowed to rise to a level which threatens to exceed the value of benefits;
- that the features of the system are not reduced to a level that the expected benefits cannot be realized;
- that the delivery date is not delayed so that there is an unacceptable loss of benefits.

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

All these requirements must be consistent with the business case.

1.14 Management control

In general, management 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 1.4 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.

Exercise 1.9 An IT project is to replace locally held paper-based records with a centrally organized database. 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. The system cannot be properly operational until the last record has been transferred. The new system will only be successful if new transactions can be processed within certain time cycles.

Identify the data that you would collect to ensure that during execution of the project things were going to plan.

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 our example, the project management might examine the 'estimated completion date' for completing data transfer for each branch. These can be checked against 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 will fail to complete the transfer of details in time. They would then need to consider what to do (this is represented in Figure 1.4 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 of moving staff from particular branches. This is *modelling* the consequences of a potential solution. 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 could still be behind in transferring details. The reason why the branch has got behind in transferring details may be 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 inputting will not have accelerated data transfer.

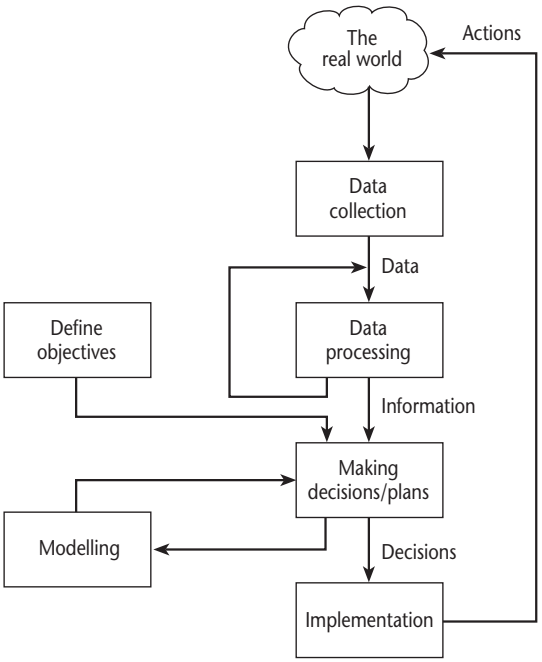

Figure 1.4 The project control cycle

It can be seen that a project plan is dynamic and will need constant adjustment during the execution of the project.

1.15 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, e.g. 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.
- Where projects involve many different people, effective channels of information have to be established. Having objective measures of success helps unambiguous communication between the various parties to a project.

Annex 1 Contents list for a feasibility study

- Introduction: identifies what the document is;
- Description of current situation;
- Problem description;
- Proposed development
  - business and financial aspects
  - technical aspects
  - organizational aspects;
- Estimated costs
  - development costs
  - operational costs;
- Envisaged benefits;
- Recommendation.

Annex 2 Contents list for a project plan

- Introduction;
- Background: including reference to the business case;
- Project objectives;
- Constraints: these could be included with project objectives;

The detail that goes into these sections will be explained in later chapters. For example, Chapter 7 relates to risk while Chapter 12 explains aspects of the management of quality.

- Project products: both deliverable products that the client will receive and intermediate products;
- Methods;
- Activities to be carried out;
- Resources to be used;
- Risks to the project;
- Management of the project, including
  - organizational responsibilities
  - management of quality
  - configuration management.

1.16 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 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 software house has developed a customized order processing system for a client. You are an employee of the software house that has been asked to organize a training course for the end-users of the system. At present, a user handbook has been produced, but no specific training material. A plan is now needed for the project which will set up the delivery of the training courses. The project can be assumed to have been completed when the first training course starts. Among the things that will need to be considered are the following:
   - Training materials will need to be designed and created;
   - A timetable will need to be drafted and agreed;
   - Date(s) for the course will need to be arranged;
   - The people attending the course will need to be identified and notified;
   - Rooms and computer facilities for the course will need to be provided for.

   (a) Identify the stakeholders for this project.
   (b) Draw up a list of the stakeholders for this project.
   (c) For each of the objectives, identify the measures of effectivness.
   (d) For each objective, write down the stakeholders who will be responsible for the achievement of that objective.
   (e) For each objective/stakeholder pair identified in (d), draft a statement of their goal or sub-objective.
5. 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 need to be in place to control this sub-project.