# 1 Introduction to Software Project Management

**Learning Objectives**

- Define the scope of 'software project management'
- 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
- Define the success criteria for a project

## 1.1 Introduction

This textbook is about '*software* project management'. The first question is whether the management of *software* projects is really that different from that of other projects. To answer this, we need to look at some key ideas about the planning, monitoring and control of software projects. We will see that all projects are about meeting objectives. Like any other project, a software project must satisfy real needs. To do this we must identify the project's stakeholders and their objectives. Ensuring that their objectives are met is the aim of project management. However, we cannot know that a project will meet its objectives in the future unless we know the present state of the project.

## 1.2 Why is Software Project Management Important?

This book is for students of software engineering and computer science and also those studying business information systems. More technically oriented students can be impatient at having to study something which keeps them away from their code. So why is it important to become familiar with project management?

The information in this paragraph comes from a National Audit Office report, *Improving IT Procurement*, November 2004.

First, there is the question of money. A lot of money is at stake with ICT projects. In the United Kingdom during the financial year 2002–2003, the central government spent more on contracts for ICT projects than on contracts related to roads (about £2.3 billion as opposed to £1.4 billion). The biggest departmental spender was the Department for Work and Pensions, who spent over £800 million on ICT. Mismanagement of ICT projects means that there is less to spend on good things such as hospitals.

There has been some debate about the precise validity of the Standish findings but the key point about the prevalence of IT project failings remains clear.

Unfortunately, projects are not always successful. In a report published in 2003, the Standish Group in the United States analysed 13,522 projects and concluded that only a third of projects were successful; 82% of projects were late and 43% exceeded their budget.

The reason for these project shortcomings is often the management of projects. The National Audit Office in the UK, for example, among other factors causing project failure identified '*lack of skills and proven approach to project management and risk management*'.

## 1.3 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 Concise English Dictionary*, 1982.

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

The emphasis on being planned assumes we can determine how to carry out a task before we start. Yet with exploratory projects this might be difficult. Planning is in essence, thinking carefully about something before you do it – even with uncertain projects this is worth doing as long as the resulting plans are seen as provisional. Other activities, such as routine maintenance, will have been performed so many times that everyone knows exactly what to do. In these cases, planning hardly seems necessary, although procedures might be documented to ensure consistency and to help newcomers.

Programme management is often used to coordinate activities on concurrent jobs.

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

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.

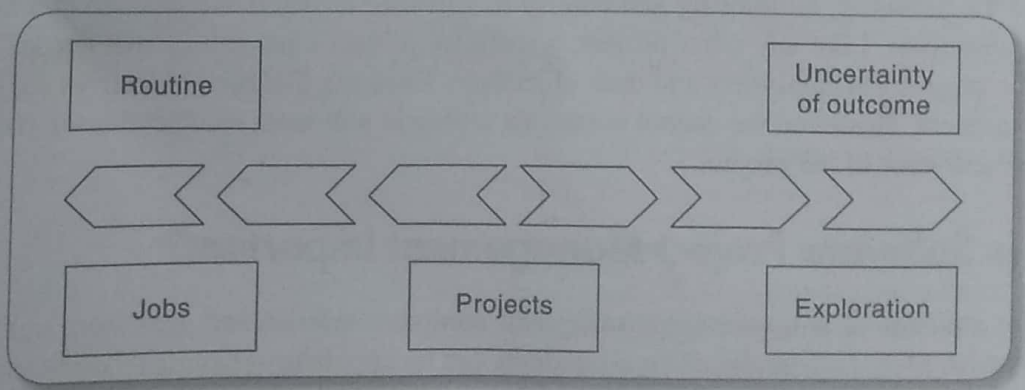



**FIGURE 1.1** Activities most likely to benefit from project management

The following characteristics distinguish projects:

- 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.
- Work is carried out for someone other than yourself.
- Work involves several specialisms.
- People are formed into a temporary work group to carry out the task.
- 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.

The more, any of these factors apply to a task, the more difficult that task will be. Project size is particularly important. The project that employs 20 developers is likely to be disproportionately more difficult than one with only 10 staff because of the need for additional coordination. The examples and exercises used in this book usually relate to smaller projects in order to make the techniques easier to grasp. However, the techniques and issues discussed are of equal relevance to larger projects.

## Exercise 1.1

Consider the following:

- Producing an edition of a newspaper
- Putting a robot vehicle on Mars to search for signs of life
- 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 seem more like real projects than others. Put them into an order most closely matching 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.

Some argue that projects are especially problematic as they are temporary sub-organizations. A group of people is brought together to carry out a task. The existence of this sub-organization cuts across the authority of the existing units within the organization. This has the advantage that a group containing various specialists is focused on a single important task. However, the project is likely to be seen as disruptive to

For example, see Rolf A. Lundin and Andres Söderholm (1995) 'A theory of the temporary organization' *Scandinavian Journal of Management* 11(4) 437–55.

others. Also, expertise built up during the project may be lost when the team is eventually dispersed at the end of the project.

## 1.4 Software Projects versus Other Types of Project

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

Many techniques in general project management also apply to software project management, but Fred Brooks identified some characteristics of software projects which make them particularly difficult:

*Invisibility* When a physical artefact such as a bridge is constructed, the progress can actually be seen. With software, progress is not immediately visible. Software project management can be seen as the process of making the invisible visible.

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

*Conformity* The 'traditional' engineer usually works with physical systems and materials like cement and steel. These physical systems have complexity, but are governed by consistent physical laws. 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'.

*Flexibility* That software is easy to change is seen as a strength. However, where the software system interfaces with a physical or organizational system, it is expected that the software will change to accommodate the other components rather than *vice versa*. Thus, software systems are particularly subject to change.

## 1.5 Contract Management and Technical Project Management

*In-house* projects are where the users and the developers of new software work for the same organization. However, increasingly organizations contract out ICT development to outside developers. Here, the client organization will often appoint a 'project manager' to supervise the contract who will delegate many technically oriented decisions to the contractors. Thus, the project manager will not worry about estimating the effort needed to write individual software components as long as the overall project is within budget and on time. On the supplier side, there will need to be project managers who deal with the more technical issues. This book leans towards the concerns of these 'technical' project managers.

## 1.6 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 'off the shelf', there may be no software writing as such, but this is still fundamentally a software project because so many of the other activities associated with software will still be present.

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

1. **The feasibility study** assesses whether a project is worth starting – that it has a valid *business case*. Information is gathered about the requirements of the proposed application. Requirements elicitation can, at least initially, be complex and difficult. The stakeholders may know the aims they wish to pursue, but not be sure about the means of achievement. The developmental and operational costs, and

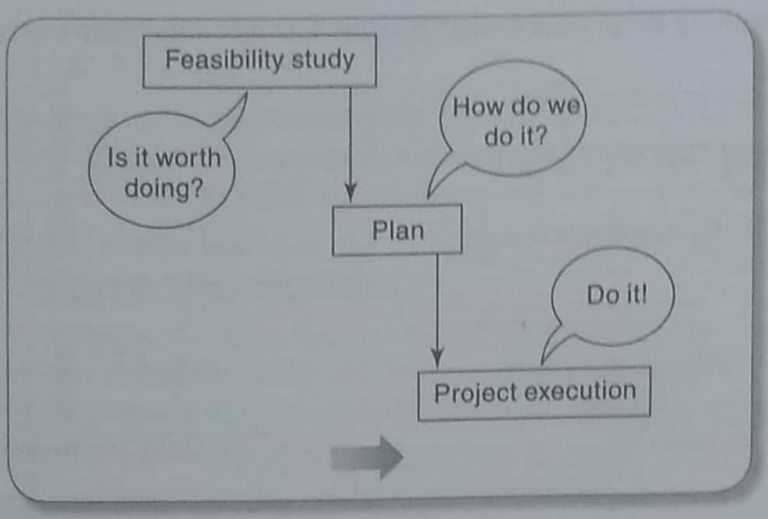



**FIGURE 1.2** The feasibility study/plan/execution cycle

the value of the benefits of the new system, will also have to be estimated. With a large system, the feasibility study could be a project in its own right with its own plan. The study could be part of a strategic planning exercise examining a range of potential software developments. Sometimes an organization assesses a *programme* of development made up of a number of projects.

Chapter 2 explores some further aspects of programme management.

2. **Planning** If the feasibility study indicates that the prospective project appears viable, then project planning can start. For larger projects, we would not do all our detailed planning at the beginning. We create an outline plan for the whole project and a detailed one for the first stage. Because we will have more detailed and accurate project information after the earlier stages of the project have been completed, planning of the later stages is left to nearer their start.

The PRINCE2 method, which is described in Appendix A, takes this iterative approach to planning. Annex 1 to this chapter has an outline of the content of a plan.

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 that the boundary between design and planning can be hazy. Design is making decisions about the form of the *products* to be created. This could relate to the external appearance of the software, that is, the user interface, or the internal architecture. The plan details the *activities* to be carried out to create these products. Planning and design can be confused because at the most detailed level, planning decisions are influenced by design decisions. Thus a software product with five major components is likely to require five sets of activities to create them.

Figure 1.3 shows the typical sequence of software development activities recommended in the international standard ISO 12207. Some activities are concerned with the *system* while others relate to *software*. The development of software will be only one part of a project. Software could be developed, for example, for a project which also requires the installation of an ICT infrastructure, the design of user jobs and user training.

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.

- *Requirements analysis* starts with *requirements elicitation* or requirements gathering which establishes what the potential users and their managers require of the new system. It could relate to a *function* – that the system should do something. It could be a quality requirement – how well the functions must work. An example of this

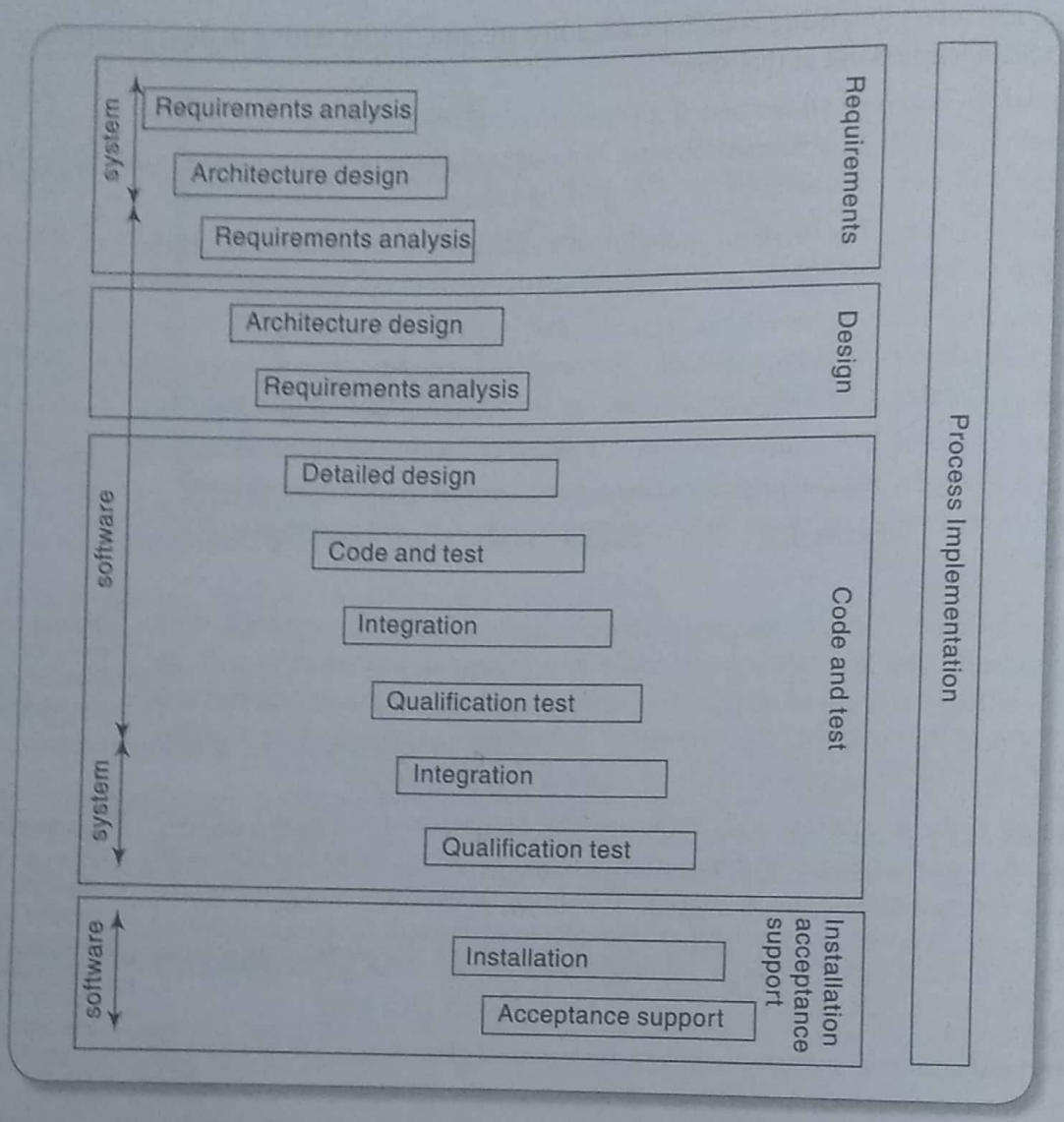



**FIGURE 1.3** The ISO 12207 software development life cycle

is dispatching an ambulance in response to an emergency telephone call. In this case transaction time would be affected by hardware and software performance as well as the speed of human operation. Training to ensure that operators use the computer system efficiently is an example of a *system requirement* for the project, as opposed to a specifically *software requirement*. There would also be *resource requirements* that relate to application development costs.

- *Architecture design* The components of the new system that fulfil each requirement have to be identified. Existing components may be able to satisfy some requirements. In other cases, a new component will have to be made. These components are not only software: they could be new hardware or work processes. Although software developers are primarily concerned with software components, it is very rare that these can be developed in isolation. They will, for example, have to take account of existing legacy systems with which they will interoperate. The design of the *system architecture* is thus an input to the *software requirements*. A second architecture design process then takes place that maps the software requirements to *software components*.
- *Detailed design* Each software component is made up of a number of software units that can be separately coded and tested. The detailed design of these units is carried out separately.

- *Code and test* refers to writing code for each software unit. Initial testing to debug individual software units would be carried out at this stage.
- *Integration* The components are tested together to see if they meet the overall requirements. Integration could involve combining different software components, or combining and testing the software element of the system in conjunction with the hardware platforms and user interactions.
- *Qualification testing* The system, including the software components, has to be tested carefully to ensure that all the requirements have been fulfilled.
- *Installation* This is the process of making the new system operational. It would include activities such as setting up standing data (for example, the details for employees in a payroll system), setting system parameters, installing the software onto the hardware platforms and user training.
- *Acceptance support* This is the resolving of problems with the newly installed system, including the correction of any errors, and implementing agreed extensions and improvements. Software maintenance can 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 outputs 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?

## Exercise 1.3

Assume that a software organization development has been asked to carry out a feasibility study to develop the payroll package for Brightmouth College. The development organization plans to develop the software by customizing one of its existing products. What are the main steps through which the project manager of the organization would carry out the feasibility study?

## 1.7 Plans, Methods and Methodologies

A plan for an activity must be based on some idea of a *method* of work. For example, if you were asked to test some software, you may know nothing about the software to be tested, but you could assume that you would need to:

- Analyse the requirements for the software.

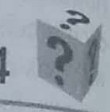
- Devise and write test cases that will check that each requirement has been satisfied.
- Create test scripts and expected results for each test case.
- Compare the actual results and the expected results and identify discrepancies.

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 – including information – will be needed

The output from one method might be the input to another. Groups of methods or techniques are often grouped into *methodologies* such as object-oriented design.

## Exercise 1.4



This should ideally be done in groups of about four, but you can think about how you would go about this exercise on your own if need be. You are probably in a building that has more than one storey. From the point of view of this exercise, the bigger the building the better.

In a group of four, work out how you would obtain an accurate estimate of the height of the building. (If you happen to be in a single-storey building, you can estimate the floor area instead!) Plan how you would carry out any action needed to obtain your estimate. Spend 20 minutes on this – you must remain in the same room for this planning phase. Once planning is complete, implement your plan, timing how long it takes to produce your final figure.

If there is more than one group carrying out this exercise, after completion of the task you can compare answers and also the approach you used when coming up with your answer.

## 1.8 Some Ways of Categorizing Software Projects

Projects may differ because of the different technical products to be created. Thus we need to identify the characteristics of a project which could affect the way in which it should be planned and managed. Other factors are discussed below.

### Changes to the characteristics of software projects

Over the last few decades, the characteristics of software projects have undergone drastic changes. For example, in the initial years of software development, almost every software was being entirely written from scratch. A possible reason for this could be that in those early days of software development, not many programs were available from which code could be reused in a new software development project. As a result, the entire code for every project had to be written from scratch. Further inhibiting code reuse was the fact that the programming paradigms that existed in those days hardly provided any support for code reuse. In contrast, at present, almost every programming language supports several ways of reusing existing code, including elegant ways of customizing and extending existing code, efficiently and dynamically linking library routines and support for frameworks.

A large number of projects that are being undertaken now are essentially customization of some existing software or development for a new release of an existing software. In these projects, much of the code of the developed software comes from reused code and only a small part of the code is actually written by the development team. Consequently, project durations have now shrunk to only a few months compared to multi-year projects that were being undertaken till about a few decades back. Further, in the past, customer participation in software projects was largely restricted to only the initial interactions for requirements, gathering, and specification and later, on completion of the project, taking delivery of the developed software. In contrast, at present, customer participation in almost every aspect of a project is being actively encouraged and often a few customer representatives are being included in the development team.

## Compulsory versus voluntary users

In workplaces, there are systems that staff have to use if they want to do something, such as recording a sale. However, use of a system is increasingly voluntary, as in the case of computer games. Here it is difficult to elicit precise requirements from potential users as we could with a business system. What the game will do will thus depend much on the informed ingenuity of the developers, along with techniques such as market surveys, focus groups and prototype evaluation.

## Information systems versus embedded systems

A traditional distinction has been between *information systems* which enable staff to carry out office processes and *embedded systems* which control machines. A stock control system would be an information system. An embedded, or process control, system might control the air conditioning equipment in a building. Some systems may have elements of both where, for example, the stock control system also controls an automated warehouse.

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

## Exercise 1.5

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

## Software products *versus* services

All types of software projects can broadly be classified into software product development projects and software services projects. These two broad classes of software projects can be further classified into subclasses, as shown in Figure 1.4. A software product development project concerns developing the software by keeping the requirements of the general customers in mind and the developed software is usually sold off-the-shelf to a large number of customers. A software product development project can further be classified depending on whether it concerns developing a generic product or a domain-specific product. A generic software product is sold to a broad spectrum of customers and is said to have a horizontal market. Examples of generic software products are Microsoft's Windows operating system and Oracle Corporation's Oracle 8i database management software. On the other hand, domain-specific software products are sold to specific categories of customers and are said to have a vertical market. Domain-specific software products target specific segments of customers (called verticals), such as, banking, telecommunication, finance and accounts and medical. Examples of domain-specific software products are BANCS from TCS and FINACLE from Infosys in the banking domain and AspenPlus from Aspen corporation in the chemical process simulation.

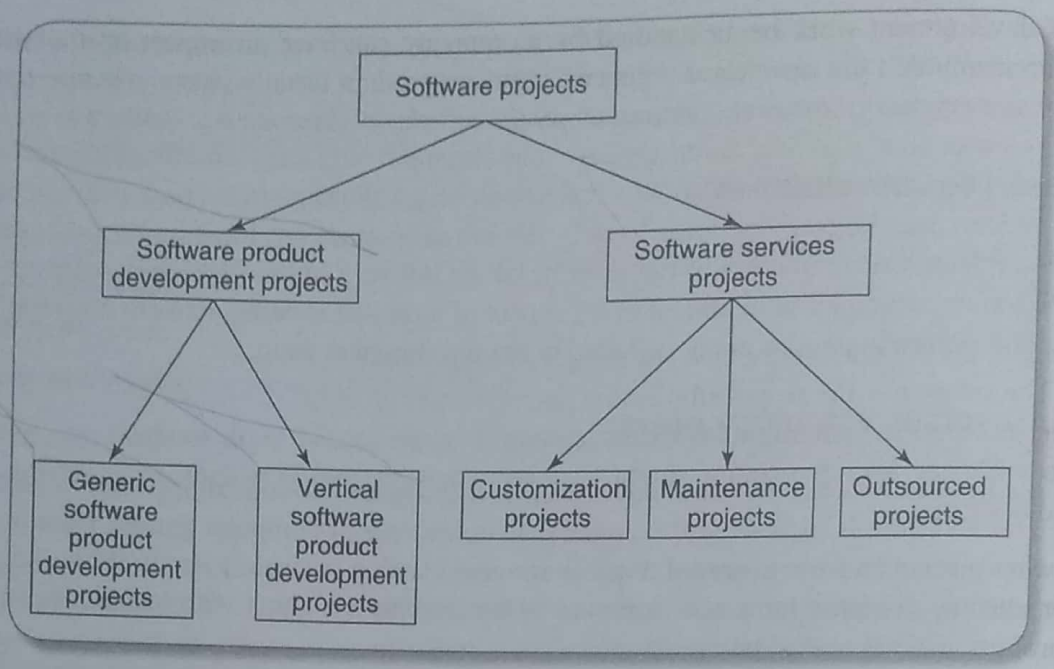



**FIGURE 1.4** A classification of software projects

Software services on the other hand, cover a large gamut of software projects such as customization, outsourcing, maintenance, testing and consultancy. At present, there is a rapid growth in the number of software services projects that are being undertaken world-wide and software services are poised to become the dominant type of software projects. One of the reasons behind this situation is the steep growth in the available code base. Over the past few decades, a large number of programs have already been developed. Available programs can therefore be modified to quickly fulfil the specific requirements of any customer. At present, there is hardly any software project in which the program code is written from scratch, and software is being mostly developed by customizing some existing software. For example, to develop a software to automate the payroll generation activities of an educational institute, the vendor may customize an existing software that might have been developed earlier for a different client's educational institute. Due to heavy reuse of code, it has now become possible to develop even large software systems in rather short period of time. Therefore, typical project durations are at present only a couple of months and multi-year projects have become very rare.

## Outsourced projects

While developing a large project, sometimes, it makes good commercial sense for a company to outsource some parts of its work to other companies. There can be several reasons behind such a decision. For example, a company may consider outsourcing as a good option, if it feels that it does not have sufficient expertise to develop some specific parts of the product or if it determines that some parts can be developed cost-effectively by another company. Since an outsourced project is a small part of some project, it is usually small in size and needs to be completed within a few months. Considering these differences between an outsourced project and a conventional project, managing an outsourced project entails special challenges.

Indian software companies excel in executing outsourced software projects and have earned a fine reputation in this field all over the world. Of late, the Indian companies have slowly begun to focus on product development as well.

The type of development work being handled by a company can have an impact on its profitability. For example, a company that has developed a generic software product usually gets an uninterrupted stream of revenue over several years. However, outsourced projects fetch only one time revenue to any company.

## Objective-driven development

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

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

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 requirement might be to meet certain objectives which could be met in a number of ways. An organization might have a problem and ask a specialist to recommend a solution.

Many software projects have two stages. First is an objective-driven project resulting in recommendations. This might identify the need for a new software system. The next stage is a project actually to create the software product. This is useful where the technical work is being done by an external group and the user needs are unclear at the outset. The external group can produce a preliminary design at a fixed fee. If the design is acceptable the developers can then quote a price for the second, implementation, stage based on an agreed requirement.

### Exercise 1.6

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

## 1.9 Project Charter

Project charter is an important high-level document that authorizes the starting of a project and use of the required resources. Besides, it outlines the project objectives, deliverables and the resources required. It also documents the aspects that are out of scope, and identifies the main stakeholders, their roles and responsibilities. A project charter document is usually developed for all types of projects, irrespective of whether it is an in-house project or a project undertaken on behalf of some customers. The project charter is signed and issued by a member of the top management of the company who also takes up the role of the sponsor of the project. The project sponsor champions the project, monitors the progress of the project and provides regular feedback to the project manager, and helps in removing any obstacles that the project manager may find difficult to overcome.

The project manager for a project is usually appointed before the project charter is issued and undertakes to write the project charter in consultation with the project sponsor. The project charter serves as a guiding document for all activities concerning the project. This is a document that is not expected to change throughout the project life cycle, unlike other project management documents such as project plan, risk management plan and work breakdown structure (WBS). The latter documents are dynamic in nature and change several times during the project duration. The project manager refers to the project charter while planning the project.

The project charter is usually a short document that is only a couple of pages long and typically contains the following:

- Overall objectives of the project and the broad items that are within the scope of the project and those that are out of scope.
- The time schedule in terms of the start date and the expected completion date of the project.
- The important stakeholders and their responsibilities towards the project.
- Overview of the resources that will be needed for the project and the overall budget.
- Major risks to the project and the broad strategies that can be adopted for overcoming those.

## 1.10 Stakeholders

These are people who have a stake or interest in the project. Their early identification is important as you need to set up adequate communication channels with them. Stakeholders can be categorized as:

- *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 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. They may be contractors who will carry out work for the project. The relationship here is usually based on a contract.

B.W. Boehm and R. Ross, 'Theory W software project management: principles and examples', in B. W. Boehm (ed.) (1989) Software Risk Management, IEEE Computer Society Press.

Different types of stakeholders may have different objectives and one of the jobs of the project leader is to recognize these different interests and to be able to reconcile them. For example, end-users may be concerned with the ease of use of the new application, while their managers may be more focused on staff savings. The project leader therefore 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 benefit from a project and therefore have an interest in its success. (The 'W' stands for 'win–win'.)

The role and format of communication plans will be explained in greater detail in Chapter 11 on managing people in software environments.

Project managers can sometimes miss an important stakeholder group, especially in unfamiliar business contexts. These could be departments supplying important services that are taken for granted.

Given the importance of coordinating the efforts of stakeholders, the recommended practice is for a *communication plan* to be created at the start of a project.

### Exercise 1.7

Identify the stakeholders in the Brightmouth College payroll project.

## 1.11 Setting Objectives

Among all these stakeholders are those who actually own the project. They control the financing of the project. They also set the objectives of the project. The objectives should define what the project team must achieve for project success. Although different stakeholders have different motivations, the project objectives identify the shared intentions for the project.

Objectives focus on the desired outcomes of the project rather than the tasks within it – they are the 'post-conditions' of the project. Informally the objectives could be written as a set of statements following the opening words '*the project will be a success if. . . .*' Thus one statement in a set of objectives might be '*customers can order our products online*' rather than '*to build an e-commerce website*'. There is often more than one way to meet an objective and the more possible routes to success the better.

There may be several stakeholders, including users in different business areas, who might have some claim to project ownership. In such a case, a *project authority* needs to be explicitly identified with overall authority over the project.

This authority is often a *project steering committee* (or *project board* or *project management board*) with overall responsibility for setting, monitoring and modifying objectives. The project manager runs the project on a day-to-day basis, but regularly reports to the steering committee.

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

### Sub-objectives and goals

An effective objective for an individual must be something that is within the control of that individual. An objective might be that the software application produced must pay for itself by reducing staff costs. As an overall business objective this might be reasonable. For software developers it would be unreasonable as any reduction in operational staff costs depends not just on them but on the operational management of the delivered system. A more appropriate *goal* or sub-objective for the software developers would be to keep development costs within a certain budget.

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

We can say that in order to achieve the objective we must achieve certain goals or sub-objectives first. These 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. Informally this can be expressed as a set of statements following the words '*To reach objective. . ., the following must be in place. . .*'.

The mnemonic SMART is sometimes used to describe well-defined objectives:

- *Specific* Effective objectives are concrete and well defined. Vague aspirations such as '*to improve customer relations*' are unsatisfactory. Objectives should be defined so that it is obvious to all whether the project has been successful.
- *Measurable* Ideally there should be *measures of effectiveness* which tell us how successful the project has been. For example, '*to reduce customer complaints*' 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 1 June?*'
- *Achievable* It must be within the power of the individual or group to achieve the objective.

This still leaves a problem about the level at which the target should be set, e.g. why, say, a 50% reduction in complaints and not 40% or 60%?

- *Relevant* The objective must be relevant to the true purpose of the project.
- *Time constrained* There should be a defined point in time by which the objective should have been achieved.

## Exercise 1.8

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 the fewest possible software errors 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

These concepts are explained more fully in Chapter 13 on software quality.

Measures of effectiveness provide practical methods of checking that an objective has been met. 'Mean time between failures' (mtbf) might, for example, be used to measure reliability. This is a *performance* measurement and, as such, can only be taken once the system is operational. Project managers want to get some idea of the performance of the completed system as it is being constructed. They will therefore seek *predictive* measures. For example, a large number of errors found during code inspections might indicate potential problems with reliability later.

## Exercise 1.9

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?

## 1.12 The Business Case

The business case should be established at the time of the project's feasibility study. Chapter 2 explains the idea of a business case in more detail.

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 where the expected benefits cannot be realized;
- that the delivery date is not delayed so that there is an unacceptable loss of benefits.

## 1.13 Project Success and Failure

The project plan should be designed to ensure project success by preserving the business case for the project. However, every non-trivial project will have problems, and at what stage do we say that a project is actually a failure? Because different stakeholders have different interests, some stakeholders in a project might see it as a success while others do not.

A good introduction to the issues discussed here can be found in A. J. Shenhar and O. Levy (1997) 'Mapping the dimensions of project success' *Project Management Journal* 28(2) 9–12.

Broadly speaking, we can distinguish between *project objectives* and *business objectives*. The project objectives are the targets that the project team is expected to achieve. In the case of software projects, they can usually be summarized as delivering:

- The agreed functionality
- The required level of quality
- On time
- Within budget

A project could meet these targets but the application, once delivered could fail to meet the business case. A computer game could be delivered on time and within budget, but might then not sell. A commercial website used for online sales could be created successfully, but customers might not use it to buy products, because they could buy the goods more cheaply elsewhere.

We have seen that in business terms it can generally be said that a project is a success if the value of benefits exceeds the costs. We have also seen that while project managers have considerable control over development costs, the value of the benefits of the project deliverables is dependent on external factors such as the number of customers. Project objectives still have some bearing on eventual business success. As we will see in Chapter 2, increasing development costs reduce the chances of the delivered product being profitable. A delay in completion reduces the amount of time during which benefits can be generated and diminishes the value of the project.

The assessment of the value of project benefits is explored in greater depth in Chapter 2.

A project can be a success on delivery but then be a business failure, On the other hand, a project could be late and over budget, but its deliverables could still, over time, generate benefits that outweigh the initial expenditure.

Some argue that the possible gap between project and business concerns can be reduced by having a broader view of projects that includes business issues. For example, the project management of an e-commerce website implementation could plan activities such as market surveys, competitor analysis, focus groups, prototyping, and evaluation by typical potential users – all designed to reduce business risks.

For a wider discussion of the relationships between successive projects, see M. Engwall (2003) 'No project is an island: linking projects to history and context' Research Policy 32 789–808.

Because the focus of project management is, not unnaturally, on the immediate project, it may not be seen that the project is actually one of a sequence. Later projects benefit from the technical skills learnt on earlier projects. Technical learning will

increase costs on the earlier projects, but later projects benefit as the learnt technologies can be deployed more quickly, cheaply and accurately. This expertise is often accompanied by additional software assets, for example, reusable code. Where software development is outsourced, there may be immediate savings, but these longer-term benefits of increased expertise will be lost. Astute managers may assess which areas of technical expertise it would be beneficial to develop.

*Customer relationships* can also be built up over a number of projects. If a client has trust in a supplier who has done satisfactory work in the past, they are more likely to use that company again, particularly if the new requirement builds on functionality already delivered. It is much more expensive to acquire new clients than it is to retain existing ones.

## 1.14 What is Management?

We have explored some of the special characteristics of software. We now look at the 'management' aspect of software project management. It has been suggested that management involves the following activities:

- 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 clients, users, developer, suppliers and other stakeholders

### Exercise 1.10

Paul Duggan is the manager of a software development section. On Tuesday at 10.00 a.m. 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 p.m. 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 human resources 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.

Much of the project manager's time is spent on only three of the eight identified activities, *viz.*, project planning, monitoring, and control. The time period during which these activities are carried out is indicated in Figure 1.5. It shows that project management is carried out over three well-defined stages or processes,

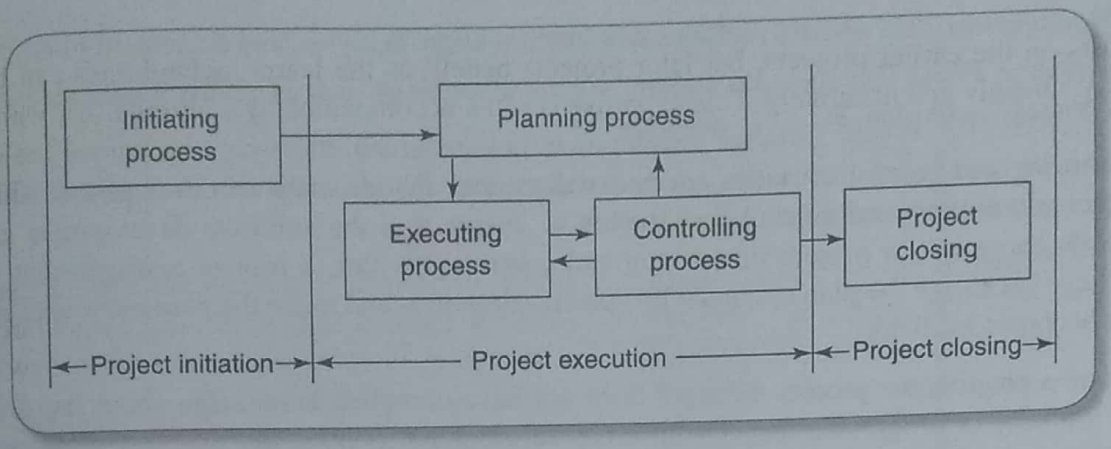



**FIGURE 1.5** Principal project management processes

irrespective of the methodology used. In the project initiation stage, an initial plan is made. As the project starts, the project is monitored and controlled to proceed as planned. However, the initial plan is revised periodically to accommodate additional details and constraints about the project as they become available. Finally, the project is closed. In the project closing stage, all activities are logically completed and all contracts are formally closed.

Initial project planning is undertaken immediately after the feasibility study phase and before starting the requirements analysis and specification process. Figure 1.5 shows this project initiation period. Initial project planning involves estimating several characteristics of a project. Based on these estimates, all subsequent project activities are planned. The initial project plans are revised periodically as the project progresses and more project data becomes available. Once the project execution starts, monitoring and control activities are taken up to ensure that the project execution proceeds as planned. The monitoring activity involves monitoring the progress of the project. Control activities are initiated to minimize any significant variation in the plan.

Project planning is an important responsibility of the project manager. During project planning, the project manager needs to perform a few well-defined activities that have been outlined below. Note that we have given a very brief description of these activities in this chapter. We will discuss these activities in more detail in subsequent chapters. Several best practices have been proposed for software project planning activities. In Chapter 3, we will discuss Step Wise, which is based on the popular PRINCE2 (PRojects IN Controlled Environments) method. While PRINCE2 is used extensively in the UK and Europe, similar software project management best practices have been put forward in the USA by the Project Management Institute's 'PMBOK' which refers to their publication 'A Guide to the Project Management Body of Knowledge.'

- *Estimation* The following project attributes are estimated.
- *Cost* How much is it going to cost to complete the project?
- *Duration* How long is it going to take to complete the project?
- *Effort* How much effort would be necessary for completing the project?

The effectiveness of all activities such as scheduling and staffing, which are planned at a later stage, depends on the accuracy with which the above three project parameters have been estimated.

- *Scheduling* Based on estimations of effort and duration, the schedules for manpower and other resources are developed.
- *Staffing* Staff organization and staffing plans are made.

- *Risk Management* This activity includes risk identification, analysis, and abatement planning.
- *Miscellaneous Plans* This includes making several other plans such as quality assurance plan, configuration management plan, etc.

Project monitoring and control activities are undertaken after the initiation of development activities. The aim of project monitoring and control activities is to ensure that the software development proceeds as planned. While carrying out project monitoring and control activities, a project manager may sometimes find it necessary to change the plan to cope with specific situations and make the plan more accurate as more project data becomes available.

At the start of a project, the project manager does not have complete knowledge about the details of the project. As the project progresses through different development phases, the manager's information base gradually improves. The complexities of different project activities become clear, some of the anticipated risks get resolved, and new risks appear. The project parameters are re-estimated periodically incorporating new understanding and change in project parameters. By taking these developments into account, the project manager can plan subsequent activities more accurately with increasing levels of confidence. Figure 1.5 shows this aspect as iterations between monitoring and control, and the plan revision activities.

## 1.15 Management Control

Management, in general, involves setting objectives for a system and then monitoring the performance of the system. In Figure 1.6, 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 1.11

An ICT 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 will then have to use the new ICT system to set up the backlog 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 1.6 by the box *Making decisions/ plans*). One possibility would be to move staff temporarily from one branch to another. If this is done, there

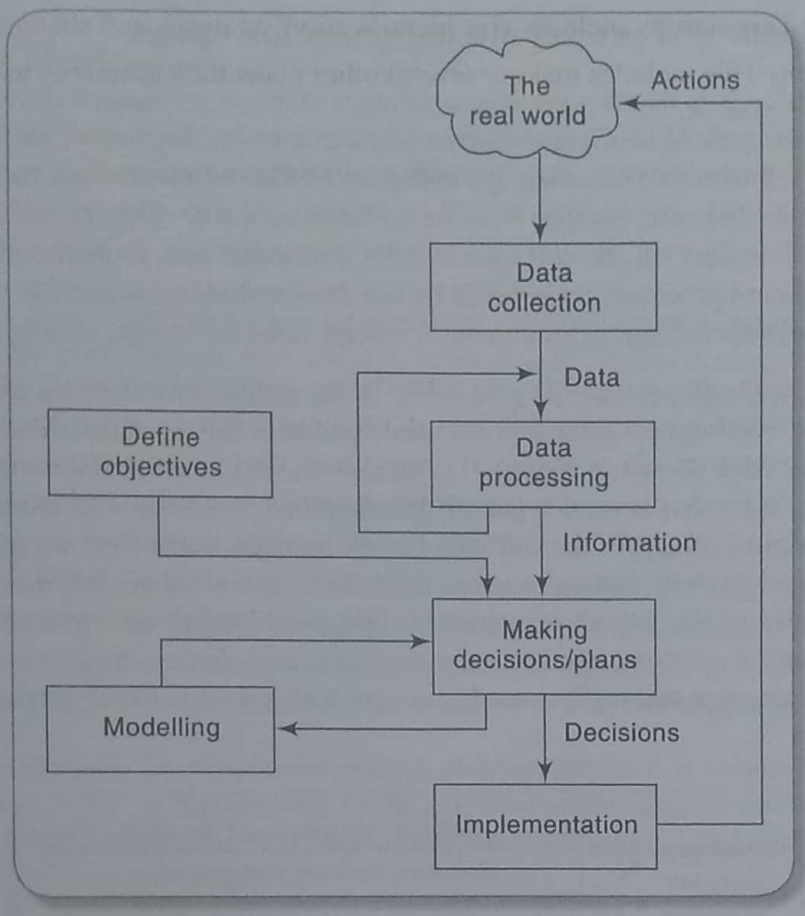



**FIGURE 1.6** The project control cycle

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. 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. 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 inputting will not have accelerated data transfer.

It can be seen that a project plan is dynamic and will need constant adjustment during the execution of the project. Courses and books on project management (such as this one) often focus considerable attention on project planning. While this is to be expected, with nearly all projects much more time is spent actually doing the project rather than planning it. A good plan provides a foundation for a good project, but is nothing without intelligent execution. The original plan will not be set in stone but will be modified to take account of changing circumstances.

## Software development and project management life cycles

Software development life cycle denotes the stages through which a software is developed. In Figure 1.7, a software development life cycle (SDLC) is shown in terms of the set of activities that are undertaken during a typical software development project, their grouping into different phases and their sequencing. During the software development life cycle, starting from its conception, the developers carry out several processes (or development methodologies) till the software is fully developed and deployed at the client site. A few examples of the development processes undertaken by the development team include requirements gathering and analysis, requirements specification, architectural design, detailed design, coding and testing.

In contrast to the software development life cycle (SDLC), the project management life cycle typically starts well before the software development activities start and continues for the entire duration of the SDLC. This aspect has schematically been shown in Figure 1.7. As shown, during the software development life cycle, the software developers carry out several types of development processes. On the other hand, during the software project management life cycle, the software project manager carries out several project management processes (or project management methodologies) to perform the required software project management activities. A few examples of the project management processes carried out by a project manager include project initiation, planning, execution, monitoring, controlling, and closing. As shown in Figure 1.7, 'project life cycle' is a more generic term and is often used to denote both software development life cycle and project management life cycle.

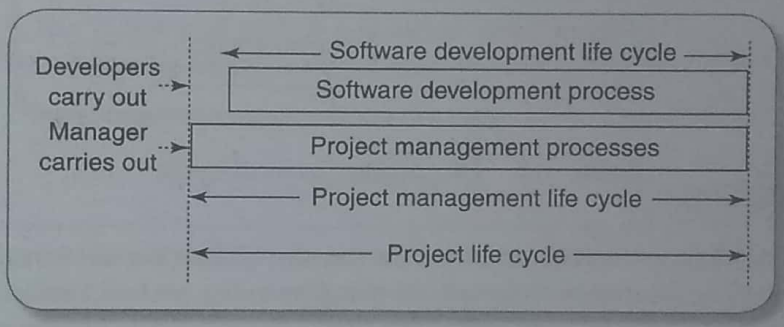



**FIGURE 1.7** Project management life cycle versus software development life cycle

The activities carried out by the developers during software development life cycle as well as the project management life cycle are grouped into a number of phases. Typical sets of phases and their sequencing in the software development life cycle and the project management life cycle have been shown in Figure 1.8. As can be seen from Figure 1.8, the phases of the software development life cycle are requirements analysis, design, development, test and delivery. The different phases of the project management life cycle are initiation, planning, execution and closing. Further observe from Figure 1.8 that by the time the software development processes start, the initiation phase of the software project management life cycle is almost complete.

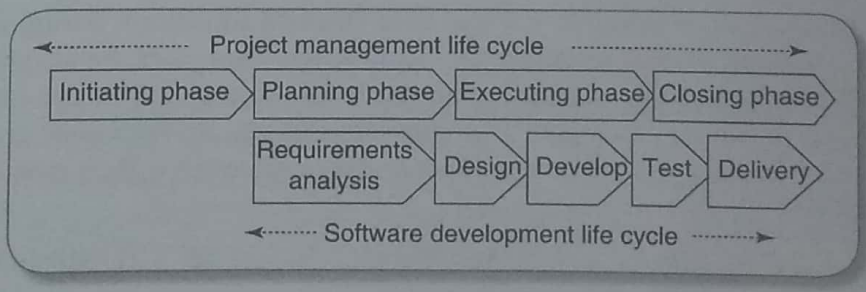



**FIGURE 1.8** Different phases of project management life cycle and software development life cycle

## 1.16 Project Management Life Cycle

The different phases of the project management life cycle are shown in Figure 1.8. In the following, we discuss the main activities that are carried out in each phase.

### Project Initiation

As shown in Figure 1.8, the software project management life cycle starts with project initiation. The project initiation phase usually starts with project concept development. During concept development the different characteristics of the software to be developed are thoroughly understood. The different aspects of the project that are investigated and understood include: the scope of the project, project constraints, the cost that would be incurred and the benefits that would accrue. Based on this understanding, a feasibility study is undertaken to determine whether the project would be financially and technically feasible. This is true for all types of projects, including the in-house product development projects as well as the outsourced projects. For example, an organization might feel a need for a software to automate some of its activities, possibly for more efficient operation. Based on the feasibility study, the business case is developed. Once the top management agrees to the business case, the project manager is appointed, the project charter is written, and finally the project team is formed. This sets the ground for the manager to start the project planning phase.

As has already been pointed out, during the project initiation phase it is crucial for the champions of the project to develop a thorough understanding of the important characteristics of the project. In his W5HH principle, Barry Boehm summarized the questions that need to be asked and answered in order to have an understanding of these project characteristics.

**W5HH Principle:** Boehm suggested that during project initiation, the project champions should have comprehensive answers to a set of key questions pertaining to the project. The answers to these questions would lead to the definition of key project characteristics. The name of this principle (W5HH) is an acronym constructed from the first letter of each question. This set of seven questions is the following:

- Why is the software being built?
- What will be done?
- When will it be done?
- Who is responsible for a function?
- Where are they organizationally located?
- How will the job be done technically and managerially?
- How much of each resource is needed?

**Project bidding:** Once an organization's top management is convinced by the business case, the project charter is developed. For some categories of projects, it may be necessary to have a formal bidding process to select a suitable vendor based on some cost-performance criteria. If the project involves automating some activities of an organization, the organization may either decide to develop it in-house or may get various software vendors to bid for the project. We briefly mention the different types of bidding techniques and their implications and applicability.

- *Request for quotation (RFQ)* An organization advertises an RFQ if it has good understanding of the project and the possible solutions. While publishing the RFQ, the organization would have to mention the scope of the work in a statement of work (SOW) document. Based on the RFQ different vendors can submit their quotations. The RFQ issuing organization can select a vendor based on the price quoted as

well as the competency of the vendor. In government organizations, the term request for tender (RFT) is usually used in place of RFQ. RFT is similar to RFQ; however, in RFT the bidder needs to deposit a tender fee in order to participate in the bidding process.

- *Request for proposal (RFP)* Many times it so happens that an organization has reasonable understanding of the problem to be solved, however it does not have a good grasp of the solution aspects. That is, the organization may not have sufficient knowledge about the different features that are to be implemented, and may lack familiarity with the possible choices of the implementation environment, such as, databases, operating systems, client-server deployment, etc. In this case, the organization may solicit solution proposals from vendors. The vendors may submit a few alternative solutions and the approximate costs for each solution. In order to develop a better understanding, the requesting organization may ask the vendors to explain or demonstrate their solutions. It needs to be understood that the purpose of RFP is to get an understanding of the alternative solutions possible that can be deployed and not vendor selection. Based on the RFP process, the requesting organization can form a clear idea of the project solutions required, based on which it can form a statement work (SOW) for requesting RFQ from the vendors.
- *Request for Information (RFI)* An organization soliciting bids may publish an RFI. Based on the vendor response to the RFI, the organization can assess the competencies of the vendors and shortlist the vendors who can bid for the work. However, it must be noted that vendor selection is seldom done based on RFI, but the RFI response from the vendors may be used in conjunction with RFP and RFQ responses for vendor selection.

## Project planning

An important outcome of the project initiation phase is the project charter. During the project planning phase, the project manager carries out several processes and creates the following documents:

- *Project plan* This document identifies the project tasks, and a schedule for the project tasks that assigns project resources and time frames to the tasks.
- *Resource plan* It lists the resources, manpower and equipment that would be required to execute the project.
- *Financial plan* It documents the plan for manpower, equipment and other costs.
- *Quality plan* Plan of quality targets and control plans are included in this document.
- *Risk plan* This document lists the identification of the potential risks, their prioritization and a plan for the actions that would be taken to contain the different risks.

## Project execution

In this phase the tasks are executed as per the project plan developed during the planning phase. A series of management processes are undertaken to ensure that the tasks are executed as per plan. Monitoring and control processes are executed to ensure that the tasks are executed as per plan and corrective actions are initiated whenever any deviations from the plan are noticed. However, the project plan may have to be revised periodically to accommodate any changes to the project plan that may arise on account of change requests, risks and various events that occur during the project execution. Quality of the deliverables is ensured through execution of proper processes. Once all the deliverables are produced and accepted by the customer, the project execution phase completes and the project closure phase starts.

### Project closure

Project closure involves completing the release of all the required deliverables to the customer along with the necessary documentation. Subsequently, all the project resources are released and supply agreements with the vendors are terminated and all the pending payments are completed. Finally, a postimplementation review is undertaken to analyse the project performance and to list the lessons learnt for use in future projects.

## 1.17 Traditional versus Modern Project Management Practices

Over the last two decades, the basic approach taken by the software industry to develop software has undergone a radical change. Hardly any software is being developed from scratch any more. Software development projects are increasingly being based on either tailoring some existing product or reusing certain pre-built libraries. In either case, two important goals of recent life cycle models are maximization of code reuse and compression of project durations. Other goals include facilitating and accommodating client feedbacks and customer participation in project development work, and incremental delivery of the product with evolving functionalities. Change requests from customers are encouraged, rather than circumvented. Clients on the other hand, are demanding further reductions in product delivery times and costs. These recent developments have changed project management practices in many significant ways. In the following section, we will discuss some important differences between modern project management practices and traditional practices.

- *Planning Incremental Delivery* Few decades ago, projects were much simpler and therefore more predictable than the present day projects. In those days, projects were planned with sufficient detail, much before the actual project execution started. After the project initiation, monitoring and control activities were carried out to ensure that the project execution proceeded as per plan. Now, projects are required to be completed over a much shorter duration, and rapid application development and deployment are considered key strategies. The traditional long-term planning has given way to adaptive short-term planning. Instead of making a long-term project completion plan, the project manager now plans all incremental deliveries with evolving functionalities. This type of project management is often called extreme project management. Extreme project management is a highly flexible approach to project management that concentrates on the human side of project management (e.g., managing project stakeholders), rather than formal and complex planning and monitoring techniques.
- *Quality Management* Of late, customer awareness about product quality has increased significantly. Tasks associated with quality management have become an important responsibility of the project manager. The key responsibilities of a project manager now include assessment of project progress and tracking the quality of all intermediate artifacts. We will discuss quality management issues in Chapter 13.
- *Change Management* Earlier, when the requirements were signed off by the customer, any changes to the requirements were rarely entertained. Customer suggestions are now actively being solicited and incorporated throughout the development process. To facilitate customer feedback, incremental delivery models are popularly being used. Product development is being carried out through a series of product versions implementing increasingly greater functionalities. Also customer feedback is solicited on each version for incorporation. This has made it necessary for an organization to keep track of the various versions and revisions through which the product develops. Another reason for the increased importance of keeping track of the versions and revisions is the following. Application development through customization has become a popular business model. Therefore, existence of a large number of versions of a product and the need to support these by a development organization has become common.

In this context, the project manager plays a key role in product base lining and version control. This has made change management a crucial responsibility of the project manager. Change management is also known as configuration management. We will discuss change management in Chapter 9.

- *Requirements Management* In modern software development practices, there is a conscious effort to develop software such that it would, to a large extent, meet the actual requirements of the customer. A basic premise of these modern development methodologies is that at the start of a project the customers are often unable to fully visualize their exact needs and are only able to determine their actual requirements after they start using the software. From this view point, modern software development practices advocate delivery of software in increments as and when the increments are completed by the development team, and actively soliciting change requests from the customer as they use the increments of the software delivered to them. In fact, a few customer representatives are included in the development team to foster close every day interactions with the customers. Contrast this with the practice followed in older development methodologies, where the requirements had to be identified upfront and these were then 'signed off' by the customer and 'frozen' before the development could start. Change requests from the customer after the start of the project were discouraged. Consequently, at present in most projects, the requirements change frequently during the development cycle. It has, therefore, become necessary to properly manage the requirements, so that as and when there is any change in requirements, the latest and up-to-date requirements become available to all. Requirements management has therefore become a systematic process of controlling changes, documenting, analysing, tracing, prioritizing requirements and then communicating the changes to the relevant stakeholders. By the term controlling changes, we mean that every change request is well managed, and problems such as accidental overwriting of a newer document with an older document are avoided.
- *Release Management* Release management concerns planning, prioritizing and controlling the different releases of a software. For almost every software, multiple releases are made during its life cycle. Starting with an initial release, releases are made each time the code changes. There are several reasons as to why the code needs to change. These reasons include functionality enhancements, bug fixes and improved execution speed. Further, modern development processes such as the agile development processes advocate frequent and regular releases of the software to be made to the customer during the software development. Starting with the release of the basic or core functionalities of the software, more complete functionalities are made available to the customers every couple of weeks. In this context, effective release management has become important.
- *Risk Management* In modern software project management practices, effective risk management is considered very important to the success of a project. A risk is any negative situation that may arise as the project progresses and may threaten the success of the project. Every project is susceptible to a host of risks that could usually be attributed to factors such as technology, personnel and customer. Unless proper risk management is practised, the progress of the project may get adversely affected. Risk management involves identification of risks, assessment of the impacts of various risks, prioritization of the risks and preparation of risk-containment plans. We discuss various aspects of project risk management in Chapter 7.
- *Scope Management* Once a project gets underway, many requirement change requests usually arise. Some of these can be attributed to the customers and the others to the development team. As we have already mentioned, modern development practices encourage the customer to come up with change requests. While all essential changes must be carried out, the superfluous and ornamental changes must be scrupulously avoided. However, while accepting change requests, it must be remembered that the three critical project parameters: scope, schedule and project cost are interdependent and are very

intricately related. If the scope is allowed to change extensively, while strictly maintaining the schedule and cost, then the quality of the work would be the major casualty. Therefore, for every scope change request, the project managers examine whether the change request is really necessary and whether the budget and time schedule would permit it. Often, the scope change requests are superfluous. For example, an overenthusiastic project team member may suggest to add features that are not required by the customer. Such scope change requests originated by the overenthusiastic team members are called *goldplating* and should be discouraged if the project is to succeed. The customer may also initiate scope change requests that are more ornamental or at best nonessential. These serve only to jeopardize the success of the project, while not adding any perceptible value to the delivered software. Such avoidable scope change requests originated by the customer are termed as scope creep. To ensure the success of the project, the project manager needs to guard against both gold plating and scope creep.

## Exercise 1.12

Assume that the development of the pay roll package of Brightmouth College has been entrusted to an organization who would develop it by customizing one of its products. Discuss the main stages through which the organization could carry out project development?

## 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 PROJECT PLAN

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

The detail that goes into these sections will be explained in later chapters. For example, Chapter 7 relates to risk while Chapter 13 explains aspects of the management of quality.