4 Project management

Objectives

The objective of this chapter is to give an overview of software project management. When you have read this chapter you will:

- understand the differences between software project management and other types of engineering project management;
- know the principal tasks of software project managers;
- understand why project planning is essential in all software projects;
- understand how graphical representations (bar charts and activity charts) are used by project managers to represent project schedules;
- understand the process of risk management and some of the risks which may arise in software projects.

Contents

The failure of many large software projects in the 1960s and early 1970s was the first indication of the difficulties of software management. Software was delivered late, was unreliable, cost several times the original estimates and often exhibited poor performance characteristics (Brooks, 1975). These projects did not fail because managers or programmers were incompetent. On the contrary, these large, challenging projects attracted people of above average ability. The fault lay in the approach to management that was used. Management techniques derived from other engineering disciplines were applied and these were ineffective for software development.

The need for management is an important distinction between professional software development and amateur programming. We need software project management because professional software engineering is always subject to budget and schedule constraints. These are set by the organisation developing the software. The software project manager's job is to ensure that the software project meets these constraints and delivers software which contributes to the business goals.

Software managers are responsible for planning and scheduling project development. They supervise the work to ensure that it is carried out to the required standards. They monitor progress to check that the development is on time and within budget. Good management cannot guarantee project success. However, bad management usually results in project failure. The software is delivered late, costs more than originally estimated and fails to meet its requirements.

Software managers do the same kind of job as other engineering project managers. However, software engineering is distinct from other types of engineering in a number of ways which can make software management particularly difficult. Some of the differences are:

1. *The product is intangible* The manager of a shipbuilding project or of a civil engineering project can see the product being developed. If a schedule slips the effect on the product is visible. Parts of the structure are obviously unfinished. Software is intangible. It cannot be seen or touched. Software project managers cannot see progress. They rely on others to produce the documentation needed to review progress.

2. *There are no standard software processes* We do not have a clear understanding of the relationships between the software process and product types. In engineering disciplines with a long history, the process is tried and tested. The engineering process for particular types of system, such as a bridge, is well understood. Our understanding of the software process has developed significantly in the past few years. However, we still cannot predict with certainty when a particular software process is likely to cause development problems.

3. *Large software projects are often 'one-off' projects* Large software projects are usually different from previous projects. Managers, therefore, do have a large body of previous experience which can be used to reduce uncertainty in plans. Consequently, it is more difficult to anticipate problems. Furthermore, rapid technological changes in computers and communications outdate previous

experience. Lessons learned from that experience may not be transferable to new projects.

Because of these problems, it is not surprising that some software projects are late, over-budget and behind schedule. Software systems are often new and technically innovative. Engineering projects (such as new transport systems) which are innovative often also have schedule problems. Given the difficulties involved, it is perhaps remarkable that so many software projects are delivered on time and to budget!

Software project management is a huge topic and cannot be covered in a single chapter. Therefore, in this chapter, I simply introduce the subject and describe three important management activities, namely project planning, project scheduling and risk management. Later chapters (in Part 6) cover other aspects of software management, including managing people, software cost estimation and quality management.

4.1 Management activities

It is impossible to write a standard job description for a software manager. The job varies tremendously depending on the organisation and on the software product being developed. However, most managers take responsibility at some stage for some or all of the following activities:

- proposal writing
- project planning and scheduling
- project costing
- project monitoring and reviews
- personnel selection and evaluation
- report writing and presentations

The first stage in a software project may involve writing a proposal to carry out that project. The proposal describes the objectives of the project and how it will be carried out. It usually includes cost and schedule estimates. It may justify why the project contract should be awarded to a particular organisation or team.

Proposal writing is a critical task as the existence of many software organisations depends on having enough proposals accepted and contracts awarded. There can be no set guidelines for this task; proposal writing is a skill which is acquired by experience. Aron (1983) includes a discussion of this aspect of a project manager's job that is still relevant today.

Project planning is concerned with identifying the activities, milestones and deliverables produced by a project. A plan must then be drawn up to guide the development towards the project goals. Cost estimation is a related activity that is concerned

with estimating the resources required to accomplish the project plan. I cover these in more detail later in this chapter and in Chapter 23.

Project monitoring is a continuing project activity. The manager must keep track of the progress of the project and compare actual and planned progress and costs. Although most organisations have formal mechanisms for monitoring, a skilled manager can often form a clear picture of what is going on by informal discussion with project staff.

Informal monitoring can often predict potential project problems as they may reveal difficulties as they occur. For example, daily discussions with project staff might reveal a particular problem in finding some software fault. Rather than waiting for a schedule slippage to be reported, the software manager might assign some expert to the problem or might decide that it should be programmed around.

During a project, it is normal to have a number of formal, project management reviews. They are concerned with reviewing overall progress and technical development of the project and considering the project's status against the aims of the organisation commissioning the software.

The development time for a large software project may be several years. During that time, organisational objectives are almost certain to change. These changes may mean that the software is no longer required or that the original project requirements are inappropriate. Management may decide to stop software development or to change the project to accommodate the changes to the organisation's objectives.

Project managers usually have to select people to work on their project. Ideally, skilled staff with appropriate experience will be available to work on the project. However, in most cases, managers have to settle for a less than ideal project team. The reasons for this are:

1. The project budget may not cover the use of highly paid staff. Less experienced, less well-paid staff may have to be used.
2. Staff with the appropriate experience may not be available either within an organisation or externally. It may be impossible to recruit new staff to the project. Within the organisation, the best people may already be allocated to other projects.
3. The organisation may wish to develop the skills of its employees. Inexperienced staff may be assigned to a project to learn and to gain experience.

The software manager has to work within these constraints when selecting project staff. However, problems are likely unless at least one project member has some experience of the type of system being developed. Without this experience, many simple mistakes are likely to be made. I discuss team building and staff selection in Chapter 22.

The project manager is usually responsible for reporting on the project to both the client and contractor organisations. Project managers must write concise, coherent documents which abstract critical information from detailed project reports. They must be able to present this information during progress reviews. Consequently, the

ability to communicate effectively both orally and in writing is an essential skill for a project manager.

4.2 Project planning

Effective management of a software project depends on thoroughly planning the progress of the project. The project manager must anticipate problems which might arise and prepare tentative solutions to those problems. A plan, drawn up at the start of a project, should be used as the driver for the project. This initial plan should be the best possible plan given the available information. It evolves as the project progresses and better information becomes available.

A structure for a software development plan is described in section 4.2.1. As well as a project plan, managers may also have to draw up other types of plan. These are briefly described in Figure 4.1 and covered in more detail in the relevant chapter elsewhere in the book.

The pseudo-code shown in Figure 4.2 describes the project planning process for software development. It shows that planning is an iterative process which is only complete when the project itself is complete. As project information becomes available during the project, the plan must be regularly revised. The overall goals of the business are an important factor which must be considered when formulating the project plan. As these change, changes to the project plan are necessary.

The planning process starts with an assessment of the constraints (required delivery date, staff available, overall budget, etc.) affecting the project. This is carried out in conjunction with an estimation of project parameters such as its structure, size, and distribution of functions. The progress milestones and deliverables are then

Figure 4.1 Types of plan

Plan	Description
Quality plan	Describes the quality procedures and standards that will be used in a project. See Chapter 24.
Validation plan	Describes the approach, resources and schedule used for system validation. See Chapter 19.
Configuration management plan	Describes the configuration management procedures and structures to be used. See Chapter 29.
Maintenance plan	Predicts the maintenance requirements of the system, maintenance costs and effort required. See Chapter 27.
Staff development plan	Describes how the skills and experience of the project team members will be developed. See Chapter 22.

Figure 4.2 Project planning

```
Establish the project constraints
Make initial assessments of the project parameters
Define project milestones and deliverables
while project has not been completed or cancelled loop
    Draw up project schedule
    Initiate activities according to schedule
    Wait ( for a while )
    Review project progress
    Revise estimates of project parameters
    Update the project schedule
    Renegotiate project constraints and deliverables
    if ( problems arise ) then
        Initiate technical review and possible revision
    end if
end loop
```

defined. The process then enters a loop. A schedule for the project is drawn up and the activities defined in the schedule are initiated or given permission to continue. After some time (usually about 2–3 weeks), progress is reviewed and discrepancies noted. Because initial estimates of project parameters are tentative, the plan will always need to be modified.

Project managers revise the assumptions about the project as more information becomes available. They replan the project schedule. If the project is delayed, they may have to renegotiate the project constraints and deliverables with the customer. If this renegotiation is unsuccessful and the schedule cannot be met, a project technical review may be held. The objective of this review is to find some alternative approach to development which falls within the project constraints and meets the schedule.

Of course, wise project managers do not assume that all will go well. Problems of some description nearly always arise during a project. The initial assumptions and scheduling should be pessimistic rather than optimistic. There should be sufficient contingency built into the plan that the project constraints and milestones need not be renegotiated every time round the planning loop.

4.2.1 The project plan

The project plan sets out the resources available to the project, the work breakdown and a schedule for carrying out the work. In some organisations, the project plan is a single document including all the different types of plan introduced above. In other cases, the project plan is solely concerned with the development process. References to other plans are included but the plans themselves are separate.

The plan structure which I describe here is for this latter type of plan. The details of the project plan vary depending on the type of project and organisation. However, most plans should include the following sections:

1. *Introduction* This briefly describes the objectives of the project and sets out the constraints (e.g. budget, time, etc.) which affect the project management.
2. *Project organisation* This describes the way in which the development team is organised, the people involved and their roles in the team.
3. *Risk analysis* This describes possible project risks, the likelihood of these risks arising and the risk reduction strategies which are proposed. Risk management is covered in section 4.4.
4. *Hardware and software resource requirements* This describes the hardware and the support software required to carry out the development. If hardware has to be bought, estimates of the prices and the delivery schedule should be included.
5. *Work breakdown* This describes the breakdown of the project into activities and identifies the milestones and deliverables associated with each activity. Milestones and deliverables are discussed in section 4.2.2.
6. *Project schedule* This describes the dependencies between activities, the estimated time required to reach each milestone and the allocation of people to activities.
7. *Monitoring and reporting mechanisms* This describes the management reports which should be produced, when these should be produced and the project monitoring mechanisms used.

The project plan should be regularly revised during the project. Some parts, such as the project schedule, will change frequently; other parts will be more stable. A document organisation which allows for the straightforward replacement of sections should be used.

4.2.2 Milestones and deliverables

Managers need information. As software is intangible, this information can only be provided as documents that describe the state of the software being developed. Without this information, it is impossible to judge progress and cost estimates and schedules cannot be updated.

When planning a project, a series of *milestones* should be established where a milestone is an end-point of a software process activity. At each milestone, there should be a formal output, such as a report, that can be presented to management. Milestone reports need not be large documents. They may simply be a short report of achievements in a project activity. Milestones should represent the end of a distinct, logical stage in the project. Indefinite milestones such as 'Coding 80 per cent complete' which are impossible to validate are useless for project management.

A *deliverable* is a project result that is delivered to the customer. It is usually delivered at the end of some major project phase such as specification, design, etc.

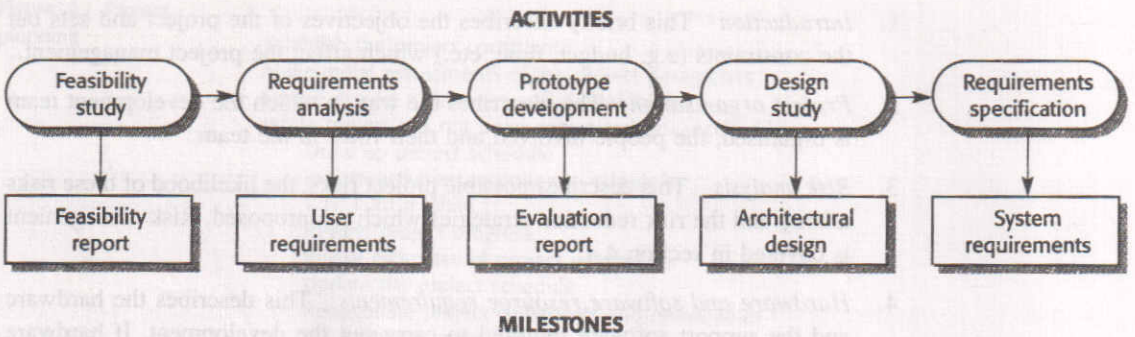


Figure 4.3 Milestones in the requirements process

Deliverables are usually milestones but milestones need not be deliverables. Milestones may be internal project results that are used by the project manager to check project progress but which are not delivered to the customer.

To establish milestones, the software process must be broken down into basic activities with associated outputs. For example, Figure 4.3 shows activities involved in requirements specification when prototyping is used to help validate requirements. The principal outputs for each activity (the project milestones) are shown. The project deliverables are the requirements definition and the requirements specification.

4.3 Project scheduling

Project scheduling is a particularly demanding task for software managers. Managers estimate the time and resources required to complete activities and organise them in a coherent sequence. Unless the project being scheduled is similar to a previous project, previous estimates are an uncertain basis for new project scheduling. Schedule estimation is further complicated by the fact that different projects may use different design methods and implementation languages.

If the project is technically advanced, initial estimates will almost certainly be optimistic even when managers try to consider all eventualities. In this respect, software scheduling is no different from scheduling any other type of large advanced project. New aircraft, bridges and even new models of cars are frequently late because of unanticipated problems. Schedules, therefore, must be continually updated as better progress information becomes available.

Project scheduling (Figure 4.4) involves separating the total work involved in a project into separate activities and judging the time required to complete these activities. Usually, some of these activities are carried out in parallel. Project schedulers must coordinate these parallel activities and organise the work so that the

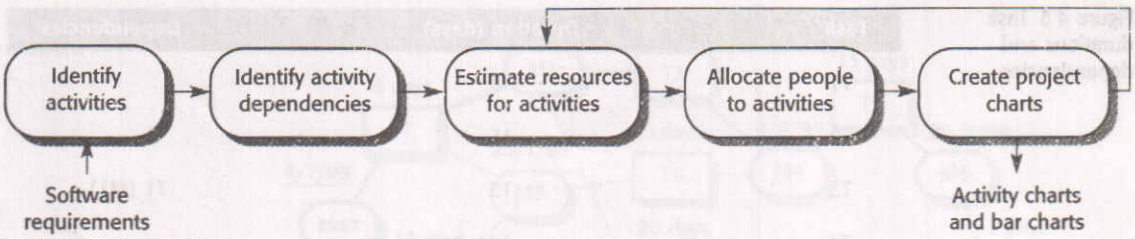


Figure 4.4 The project scheduling process

workforce is used optimally. They must avoid a situation where the whole project is delayed because a critical task is unfinished.

Project activities should normally last at least a week. Finer subdivision means that a disproportionate amount of time must be spent on estimating and chart revision. It is also useful to set a maximum amount of time for any activity of about 8 to 10 weeks. If it takes longer than this, it should be subdivided for project planning and scheduling.

In estimating schedules, managers should not assume that every stage of the project will be problem free. Individuals working on a project may fall ill or may leave, hardware may break down and essential support software or hardware may be delivered late. If the project is new and technically advanced, certain parts of it may turn out to be more difficult and take longer than originally anticipated.

As well as calendar time, managers must also estimate the resources needed to complete each task. The principal resource is the human effort required. Other resources may be the disk space required on a server, the time required on specialised hardware, such as a simulator, and the travel budget required for project staff. I discuss estimation in more detail in Chapter 23.

A good rule of thumb is to estimate as if nothing will go wrong, then increase your estimate to cover anticipated problems. A further contingency factor to cover unanticipated problems may also be added to the estimate. This extra contingency factor depends on the type of project, the process parameters (deadline, standards, etc.) and the quality and experience of the software engineers working on the project. As a rule of thumb, I always add 30 per cent to my original estimate for anticipated problems then another 20 per cent to cover other things I hadn't thought of.

The project schedule is usually represented as a set of charts showing the work breakdown, activities dependencies and staff allocations. These are discussed in the following section. Software management tools, such as Microsoft Project, are now usually used to automate chart production.

4.3.1 Bar charts and activity networks

Bar charts and activity networks are graphical notations which are used to illustrate the project schedule. Bar charts show who is responsible for each activity and when the activity is scheduled to begin and end. Activity networks show the dependencies between the different activities making up a project. Bar charts and activity

Figure 4.5 Task durations and dependencies

Task	Duration (days)	Dependencies
T1	8	
T2	15	
T3	15	T1 (M1)
T4	10	
T5	10	T2, T4 (M2)
T6	5	T1, T2 (M3)
T7	20	T1 (M1)
T8	25	T4 (M5)
T9	15	T3, T6 (M4)
T10	15	T5, T7 (M7)
T11	7	T9 (M6)
T12	10	T11 (M8)

charts can be generated automatically from a database of project information using a project management tool.

Consider the set of activities shown in Figure 4.5. This table shows activities, their duration, and activity interdependencies. From Figure 4.5, you can see that Task T3 is dependent on Task T1. This means that T1 must be completed before T3 starts. For example, T1 might be the preparation of a component design and T3, the implementation of that design. Before implementation starts, the design should be complete.

Given dependency and estimated duration of activities, an activity network which shows activity sequences may be generated (Figure 4.6). It shows which activities can be carried out in parallel and which must be executed in sequence because of a dependency on an earlier activity. Activities are represented as rectangles. Milestones and project deliverables are shown with rounded corners. Dates in this diagram show the start date of the activity and are written in British style where the day precedes the month. You should read the network from left to right and from top to bottom.

In the project management tool used to produce this chart, all activities must end in milestones. An activity may start when its preceding milestone (which may depend on several activities) has been reached. Therefore, in the third column in Figure 4.5, I have also shown the corresponding milestone (e.g. M5) which is reached when the tasks in that column finish (see Figure 4.6).

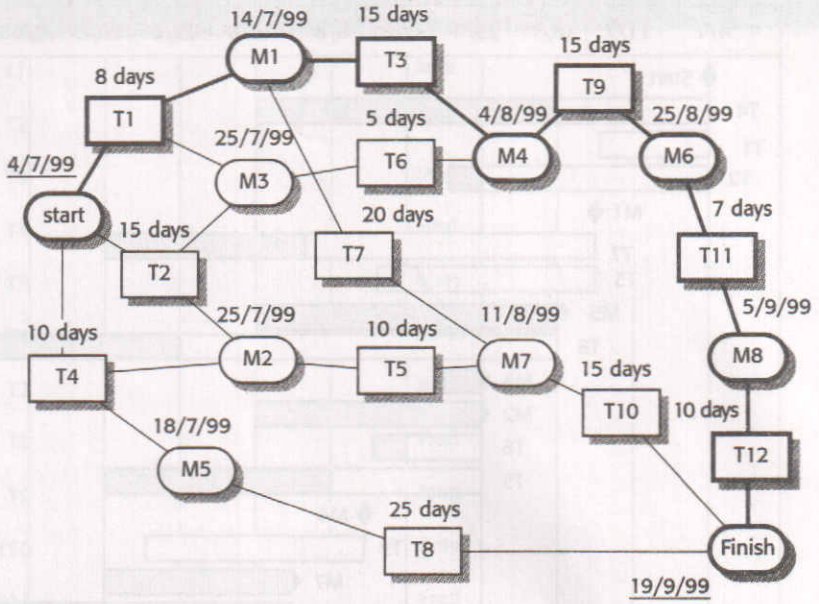


Figure 4.6 An activity network

Before progress can be made from one milestone to another, all paths leading to it must be complete. For example, task T9, shown in Figure 4.6, cannot be started until tasks T3 and T6 are finished. The arrival at milestone M4 shows that these tasks have been completed.

The minimum time required to finish the project can be estimated by considering the longest path in the activity graph (the critical path). In this case, it is 11 weeks of elapsed time or 55 working days. In Figure 4.6 the critical path is shown as a sequence of emboldened boxes. The overall schedule of the project depends on the critical path. Any slippage in the completion of any critical activity causes project delays.

Delays in activities which do not lie on the critical path, however, need not cause an overall schedule slippage. So long as the delays do not extend these activities so much that the total time exceeds the critical path, the project schedule will not be affected. For example, if T8 is delayed, it may not affect the final completion date of the project as it does not lie on the critical path. The project bar chart (Figure 4.7) shows the extent of the possible delay as a shaded bar.

Managers also use activity networks when allocating project work. They can provide insights into activity dependencies which are not intuitively obvious. It may be possible to modify the system design so that the critical path is shortened. The project schedule may be shortened because of the reduced amount of time spent waiting for activities to finish.

Figure 4.7 is an alternative way of representing project schedule information. It is a bar chart (sometimes called a Gantt chart, after its inventor) showing a project calendar and the start and finish dates of activities.

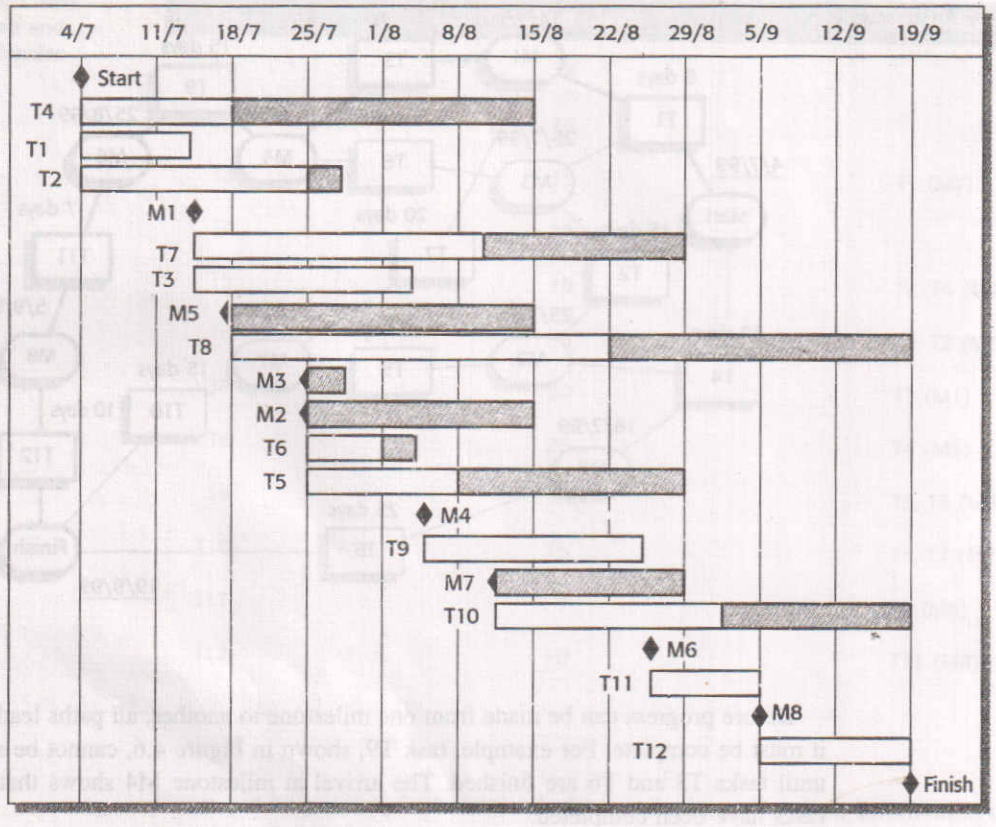


Figure 4.7 Activity bar chart

Some of the activities in Figure 4.7 are followed by a shaded bar whose length is computed by the scheduling tool. This shows that there is some flexibility in the completion date of these activities. If an activity does not complete on time, the critical path will not be affected until the end of the period marked by the shaded bar. Activities which lie on the critical path have no margin of error and they can be identified because they have no associated shaded bar.

As well as considering schedules, project managers must also consider resource allocation and, in particular, the allocation of staff to project activities. Figure 4.8 suggests an allocation of staff to the activities illustrated in Figure 4.7.

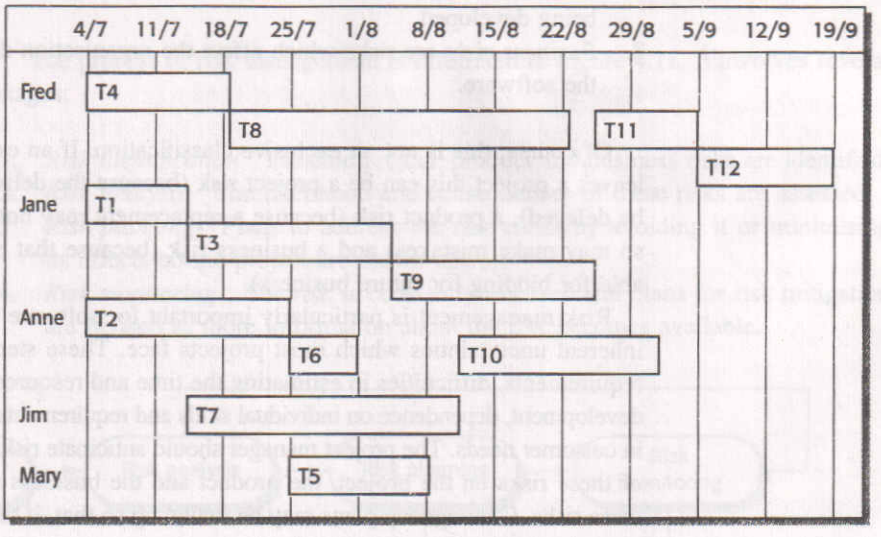
Figure 4.8 can also be processed by project management support tools and a bar chart generated which shows the time periods where staff are employed on the project (Figure 4.9). Staff don't have to be assigned to a project at all times. During intervening periods they may be on holiday, working on other projects, attending training courses or some other activity.

Large organisations usually employ a number of specialists who work on a project as required. This can cause scheduling problems. If one project is delayed while a specialist is working on it, this may have a knock-on effect on other projects. They may also be delayed because the specialist is not available.

Figure 4.8 Allocation of people to activities

Task	Engineer
T1	Jane
T2	Anne
T3	Jane
T4	Fred
T5	Mary
T6	Anne
T7	Jim
T8	Fred
T9	Jane
T10	Anne
T11	Fred
T12	Fred

Figure 4.9 Staff allocation vs time chart



Inevitably, initial project schedules will be incorrect. As a project develops, estimates should be compared with actual elapsed time. This comparison can be used as a basis for revising the schedule for later parts of the project. When actual figures are known, the activity chart should be reviewed. Later project activities may then be reorganised to reduce the length of the critical path.

4.4 Risk management

An important task of a project manager is to anticipate risks which might affect the project schedule or the quality of the software being developed and to take action to avoid these risks. The results of the risk analysis should be documented in the project plan along with an analysis of the consequences of a risk occurring. Identifying risks and drawing up plans to minimise their effect on the project is called risk management (Hall, 1998; Ould, 1999).

Simplistically, you can think of a risk as a probability that some adverse circumstance will actually occur. Risks may threaten the project, the software that is being developed or the organisation. These categories of risk can be defined as follows:

1. *Project risks* are risks which affect the project schedule or resources.
2. *Product risks* are risks which affect the quality or performance of the software being developed.
3. *Business risks* are risks which affect the organisation developing or procuring the software.

Of course, this is not an exclusive classification. If an experienced programmer leaves a project this can be a project risk (because the delivery of the system may be delayed), a product risk (because a replacement may not be as experienced and so may make mistakes) and a business risk (because that experience is not available for bidding for future business).

Risk management is particularly important for software projects because of the inherent uncertainties which most projects face. These stem from loosely defined requirements, difficulties in estimating the time and resources required for software development, dependence on individual skills and requirements changes due to changes in customer needs. The project manager should anticipate risks, understand the impact of these risks on the project, the product and the business and take steps to avoid these risks. Contingency plans may be drawn up so that, if the risks do occur, immediate recovery action is possible.

The types of risk which may affect a project depend on the project and the organisational environment where the software is being developed. However, many risks are universal and I describe some of these in Figure 4.10.

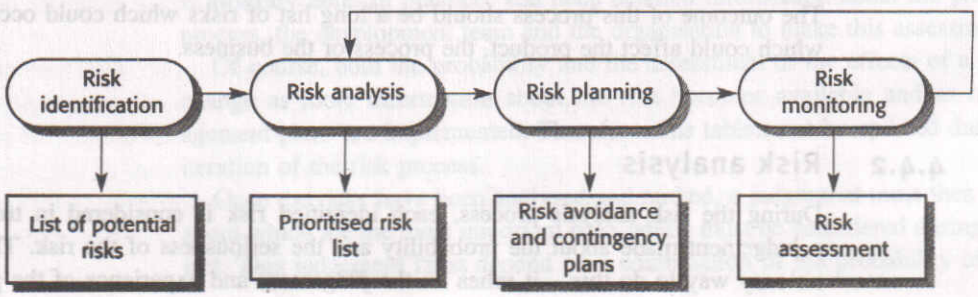
Figure 4.10 Possible software risks

Risk	Risk type	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organisational management with different priorities.
Hardware unavailability	Project	Hardware which is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool under-performance	Product	CASE tools which support the project do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

The process of risk management is illustrated in Figure 4.11. It involves several stages:

1. *Risk identification* Possible project, product and business risks are identified.
2. *Risk analysis* The likelihood and consequences of these risks are assessed.
3. *Risk planning* Plans to address the risk either by avoiding it or minimising its effects on the project are drawn up.
4. *Risk monitoring* The risk is constantly assessed and plans for risk mitigation are revised as more information about the risk becomes available.

Figure 4.11 The risk management process



The risk management process, like all other project planning, is an iterative process which continues throughout the project. Once an initial set of plans are drawn up, the situation is monitored. As more information about the risks become available, they have to be re-analysed and new priorities established. The risk avoidance and contingency plans may be modified as new risk information emerges.

The results of the risk management process should be documented in a risk management plan. This should include a discussion of the risks faced by the project, an analysis of these risks and the plans which are required to manage these risks. Where appropriate, it may also include some results of the risk management, i.e. specific contingency plans to be activated if the risk occurs.

4.4.1 Risk identification

Risk identification is the first stage of risk management. It is concerned with discovering possible risks to the project. In principle, these should not be assessed or prioritised at this stage although, in practice, risks with very minor consequences or very low probability risks are not usually considered.

Risk identification may be carried out as a team process using a brainstorming approach or may simply be based on a manager's experience. To help the process, a list of possible risk types may be used. These types include:

1. *Technology risks* Risks which derive from the software or hardware technologies which are being used as part of the system being developed.
2. *People risks* Risks which are associated with the people in the development team.
3. *Organisational risks* Risks which derive from the organisational environment where the software is being developed.
4. *Tools risks* Risks which derive from the CASE tools and other support software used to develop the system.
5. *Requirements risks* Risks which derive from changes to the customer requirements and the process of managing the requirements change.
6. *Estimation risks* Risks which derive from the management estimates of the system characteristics and the resources required to build the system.

Figure 4.12 gives some examples of possible risks in each of these categories. The outcome of this process should be a long list of risks which could occur and which could affect the product, the process or the business.

4.4.2 Risk analysis

During the risk analysis process, each identified risk is considered in turn and a judgement made about the probability and the seriousness of the risk. There is no easy way to do this – it relies on the judgement and experience of the project

Figure 4.12 Risks and risk types

Risk type	Possible risks
Technology	The database used in the system cannot process as many transactions per second as expected. Software components which should be reused contain defects which limit their functionality.
People	It is impossible to recruit staff with the skills required. Key staff are ill and unavailable at critical times. Required training for staff is not available.
Organisational	The organisation is restructured so that different management are responsible for the project. Organisational financial problems force reductions in the project budget.
Tools	The code generated by CASE tools is inefficient. CASE tools cannot be integrated.
Requirements	Changes to requirements which require major design rework are proposed. Customers fail to understand the impact of requirements changes.
Estimation	The time required to develop the software is underestimated. The rate of defect repair is underestimated. The size of the software is underestimated.

manager. These should not generally be precise numeric assessments but should be based around a number of bands:

1. The probability of the risk might be assessed as very low (<10%), low (10–25%), moderate (25–50%), high (50–75%) or very high (>75%).
2. The effects of the risk might be assessed as catastrophic, serious, tolerable or insignificant.

The results of this analysis process should then be tabulated with the table ordered according to seriousness of the risk. Figure 4.13 illustrates this for the risks identified in Figure 4.12. Obviously the assessment of probability and seriousness is arbitrary here. In practice, you need detailed information about the project, the process, the development team and the organisation to make this assessment.

Of course, both the probability and the assessment of the effects of a risk may change as more information about the risk becomes available and as risk management plans are implemented. Therefore, this table must be updated during each iteration of the risk process.

Once the risks have been analysed and ranked, a judgement must then be made about which are the most important risks which must be considered during the project. This judgement must depend on a combination of the probability of the risk

Figure 4.13 Risk analysis

Risk	Probability	Effects
Organisational financial problems force reductions in the project budget.	Low	Catastrophic
It is impossible to recruit staff with the skills required for the project.	High	Catastrophic
Key staff are ill at critical times in the project.	Moderate	Serious
Software components which should be reused contain defects which limit their functionality.	Moderate	Serious
Changes to requirements which require major design rework are proposed.	Moderate	Serious
The organisation is restructured so that different management are responsible for the project.	High	Serious
The database used in the system cannot process as many transactions per second as expected.	Moderate	Serious
The time required to develop the software is underestimated.	High	Serious
CASE tools cannot be integrated.	High	Tolerable
Customers fail to understand the impact of requirements changes.	Moderate	Tolerable
Required training for staff is not available.	Moderate	Tolerable
The rate of defect repair is underestimated.	Moderate	Tolerable
The size of the software is underestimated.	High	Tolerable
The code generated by CASE tools is inefficient.	Moderate	Insignificant

arising and the effects of that risk. In general, all catastrophic risks should always be considered, as should all serious risks which have more than a moderate probability of occurrence.

Boehm (1988) recommends identify and monitoring the 'top 10' risks but I think this figure is rather arbitrary. The right number of risks to monitor must depend on the project. It might be five or it might be 15. However, the number of risks chosen for monitoring should be manageable. A very large number of risks would simply require too much information to be collected. From the risks identified in Figure 4.13, it is appropriate to consider all eight risks which have catastrophic or serious consequences.

4.4.3 Risk planning

The risk planning process considers each of the key risks which have been identified and identifies strategies to manage the risk. Again, there is no simple process which can be followed to establish risk management plans. It relies on the judgement and experience of the project manager. Figure 4.14 shows possible strategies which have been identified for the key risks from Figure 4.13.

These strategies fall into three categories:

1. *Avoidance strategies* Following these strategies means that the probability that the risk will arise will be reduced. An example of a risk avoidance strategy is the strategy for dealing with defective components shown in Figure 4.14.
2. *Minimisation strategies* Following these strategies means that the impact of the risk will be reduced. An example of a risk minimisation strategy is the strategy for staff illness shown in Figure 4.14.
3. *Contingency plans* Following these strategies means that, if the worst happens, you are prepared for it and have a strategy in place to deal with it. An example of a contingency strategy is the strategy for organisational financial problems in Figure 4.14.

Figure 4.14 Risk management strategies

Risk	Strategy
Organisational financial problems	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Recruitment problems	Alert customer of potential difficulties and the possibility of delays, investigate buying-in components.
Staff illness	Reorganise team so that there is more overlap of work and people therefore understand each other's jobs.
Defective components	Replace potentially defective components with bought-in components of known reliability.
Requirements changes	Derive traceability information to assess requirements change impact, maximise information hiding in the design.
Organisational restructuring	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Database performance	Investigate the possibility of buying a higher-performance database.
Underestimated development time	Investigate buying-in components, investigate the use of a program generator.

Figure 4.15 Risk factors

Risk type	Potential indicators
Technology	Late delivery of hardware or support software, many reported technology problems
People	Poor staff morale, poor relationships amongst team members, job availability
Organisational	Organisational gossip, lack of action by senior management
Tools	Reluctance by team members to use tools, complaints about CASE tools, demands for higher-powered workstations
Requirements	Many requirements change requests, customer complaints
Estimation	Failure to meet agreed schedule, failure to clear reported defects

4.4.4 Risk monitoring

Risk monitoring involves regularly assessing each of the identified risks to decide whether or not that risk is becoming more or less probable and whether the effects of the risk have changed. Of course, this cannot usually be observed directly, so you have to look at other factors which give you clues about the risk probability and its effects. These factors are obviously dependent on the types of risk. Figure 4.15 gives some examples of factors which may be helpful in assessing these risk types.

Risk monitoring should be a continuous process and, at every management progress review, each of the key risks should be considered separately and discussed by the meeting.

KEY POINTS

- Good software project management is essential if software engineering projects are to be developed on schedule and within budget.
- Software management is distinct from other engineering management. Software is intangible. Projects may be novel or innovative so there is no body of experience to guide their management. Software processes are not well understood.
- Software managers have diverse roles. Their most significant activities are project planning, estimating and scheduling. Planning and estimating are iterative processes.

They continue throughout a project. As more information becomes available, plans and schedules must be revised.

- A project milestone is a predictable outcome of an activity where some formal report of progress should be presented to management. Milestones should occur regularly throughout a software project. A deliverable is a milestone which is delivered to the project customer.
- Project scheduling involves the creation of various graphical representations of part of the project plan. These include activity charts showing the interrelationships of project activities and bar charts showing activity durations.
- Major project risks should be identified and assessed to establish their probability and consequences for the project. For risks which are probable and potentially serious, plans to avoid, manage or deal with that risk when it arises should be made. Risks should be explicitly discussed at each project progress meeting.

FURTHER READING

Managing Software Quality and Business Risk. Chapter 3 of this book is simply the best discussion of risk that I have seen anywhere. The book is oriented around risk and I think it is probably the best book on this topic that is available at the time of writing. (M. Ould, 1999, John Wiley and Sons.)

The Mythical Man Month. The problems of software management have been unchanged since the 1960s and this is one of the best books on the topic. An interesting and readable account of the management of one of the first very large software projects, the IBM OS/360 operating system. The second edition includes other classic papers by Brooks. (F. P. Brooks, 1975, Addison-Wesley.)

Assessment and Control of Software Risks. There are several books on risk management now available. This one is a particularly comprehensive discussion of the types of risk, why they arise and how to avoid them. (C. Jones, 1994, Prentice-Hall.)

Principles of Software Engineering Management. This is an idiosyncratic account of software management but it contains good advice. It is written in an easy-to-read way. (T. Gilb, 1988, Addison-Wesley.)

See Part 6 for other readings on management.

EXERCISES

4.1 Explain why the intangibility of software systems poses special problems for software project management.

4.2 Explain why the best programmers do not always make the best software managers. You may find it helpful to base your answer on the list of management activities given in section 4.1.

4.3 Explain why the process of project planning is an iterative one and why a plan must be continually reviewed during a software project.

4.4 Briefly explain the purpose of each of the sections in a software project plan.

4.5 What is the critical distinction between a milestone and a deliverable?

4.6 Figure 4.16 sets out a number of activities, durations and dependencies. Draw an activity chart and a bar chart showing the project schedule.

Figure 4.16 Task durations and dependencies

Task	Duration (days)	Dependencies
T1	10	
T2	15	T1
T3	10	T1, T2
T4	20	
T5	10	
T6	15	T3, T4
T7	20	T3
T8	35	T7
T9	15	T6
T10	5	T5, T9
T11	10	T9
T12	20	T10
T13	35	T3, T4
T14	10	T8, T9
T15	20	T12, T14
T16	10	T15

4.7 Figure 4.5 gives task durations for software project activities. Assume that a serious, unanticipated setback occurs and instead of taking 10 days, task T5 takes 40 days. Revise the activity network accordingly, highlighting the new critical path. Draw up new bar charts showing how the project might be reorganised.

4.8 Using reported instances of project problems in the literature, list management difficulties which occurred in these failed programming projects. (Start with Brooks's book, as suggested in Further Reading.)

4.9 In addition to the risks shown in Figure 4.12, identify six other possible risks which are likely to arise in software projects.

4.10 You are asked by your manager to deliver software to a schedule which you know can only be met by asking your project team to work unpaid overtime. All team members have young children. Discuss whether you should accept this demand from your manager or whether you should persuade your team to give their time to the organisation rather than their families. What factors might be significant in your decision?

4.11 As a programmer, you are offered promotion to project management but you feel that you can make a more effective contribution in a technical rather than a managerial role. Discuss whether you should accept the promotion.