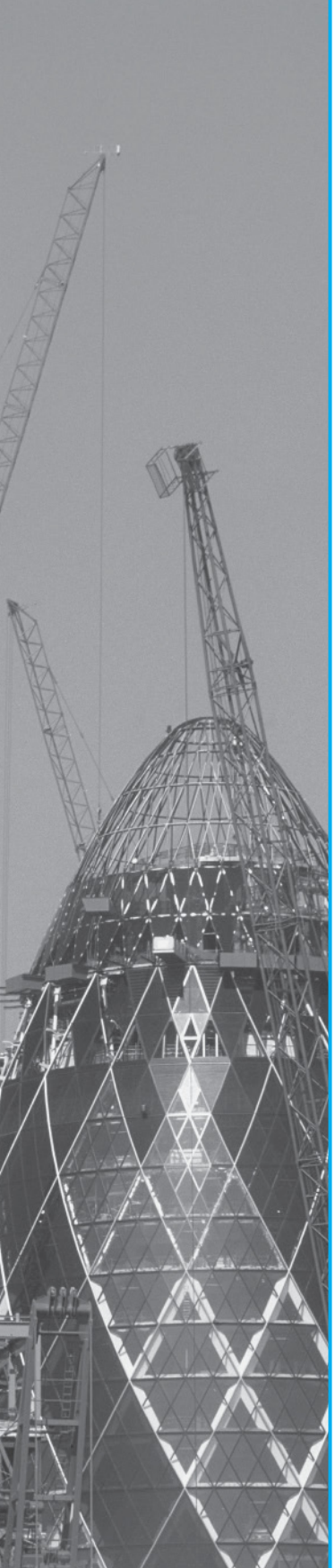

22

Project management

Objectives

The objective of this chapter is to introduce software project management and two important management activities, namely, risk management and people management. When you have read the chapter you will:

- know the principal tasks of software project managers;
- have been introduced to the notion of risk management and some of the risks that can arise in software projects;
- understand factors that influence personal motivation and what these might mean for software project managers;
- understand key issues that influence team working, such as team composition, organization, and communication.

Contents

Software project management is an essential part of software engineering. Projects need to be managed because professional software engineering is always subject to organizational budget and schedule constraints. The project manager's job is to ensure that the software project meets and overcomes these constraints as well as delivering high-quality software. Good management cannot guarantee project success. However, bad management usually results in project failure: The software may be delivered late, cost more than originally estimated, or fail to meet the expectations of customers.

The success criteria for project management obviously vary from project to project, but, for most projects, important goals are:

- to deliver the software to the customer at the agreed time;
- to keep overall costs within budget;
- to deliver software that meets the customer's expectations;
- to maintain a coherent and well-functioning development team.

These goals are not unique to software engineering but are the goals of all engineering projects. However, software engineering is different from other types of engineering in a number of ways that make software management particularly challenging. Some of these differences are:

1. *The product is intangible* A manager of a shipbuilding or 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 by looking at the artifact that is being constructed. Rather, they rely on others to produce evidence that they can use to review the progress of the work.
2. *Large software projects are often "one-off" projects* Every large software development project is unique because every environment where software is developed is, in some ways, different from all others. Even managers who have a large body of previous experience may find it difficult to anticipate problems. Furthermore, rapid technological changes in computers and communications can make experience obsolete. Lessons learned from previous projects may not be readily transferable to new projects.
3. *Software processes are variable and organization-specific* The engineering process for some types of system, such as bridges and buildings, is well understood. However, different companies use quite different software development processes. We cannot reliably predict when a particular software process is likely to lead to development problems. This is especially true when the software project is part of a wider systems engineering project or when completely new software is being developed.

Because of these issues, it is not surprising that some software projects are late, overbudget, and behind schedule. Software systems are often new, very complex, and technically innovative. Schedule and cost overruns are also common in other

engineering projects, such as new transport systems, that are complex and innovative. Given the difficulties involved, it is perhaps remarkable that so many software projects are delivered on time and to budget.

It is impossible to write a standard job description for a software project manager. The job varies tremendously depending on the organization and the software being developed. Some of the most important factors that affect how software projects are managed are:

1. *Company size* Small companies can operate with informal management and team communications and do not need formal policies and management structures. They have less management overhead than larger organizations. In larger organizations, management hierarchies, formal reporting and budgeting, and approval processes must be followed.
2. *Software customers* If the customer is an internal customer (as is the case for software product development), then customer communications can be informal and there is no need to fit in with the customer's ways of working. If custom software is being developed for an external customer, agreement has to be reached on more formal communication channels. If the customer is a government agency, the software company must operate according to the agency's policies and procedures, which are likely to be bureaucratic.
3. *Software size* Small systems can be developed by a small team, which can get together in the same room to discuss progress and other management issues. Large systems usually need multiple development teams that may be geographically distributed and in different companies. The project manager has to coordinate the activities of these teams and arrange for them to communicate with each other.
4. *Software type* If the software being developed is a consumer product, formal records of project management decisions are unnecessary. On the other hand, if a safety-critical system is being developed, all project management decisions should be recorded and justified as these may affect the safety of the system.
5. *Organizational culture* Some organizations have a culture that is based on supporting and encouraging individuals, while others are group focused. Large organizations are often bureaucratic. Some organizations have a culture of taking risks, whereas others are risk averse.
6. *Software development processes* Agile processes typically try to operate with "lightweight" management. More formal processes require management monitoring to ensure that the development team is following the defined process.

These factors mean that project managers in different organizations may work in quite different ways. However, a number of fundamental project management activities are common to all organizations:

1. *Project planning* Project managers are responsible for planning, estimating, and scheduling project development and assigning people to tasks. They supervise

the work to ensure that it is carried out to the required standards, and they monitor progress to check that the development is on time and within budget.

2. *Risk management* Project managers have to assess the risks that may affect a project, monitor these risks, and take action when problems arise.
3. *People management* Project managers are responsible for managing a team of people. They have to choose people for their team and establish ways of working that lead to effective team performance.
4. *Reporting* Project managers are usually responsible for reporting on the progress of a project to customers and to the managers of the company developing the software. They have to be able to communicate at a range of levels, from detailed technical information to management summaries. They have to write concise, coherent documents that abstract critical information from detailed project reports. They must be able to present this information during progress reviews.
5. *Proposal writing* The first stage in a software project may involve writing a proposal to win a contract to carry out an item of work. The proposal describes the objectives of the project and how it will be carried out. It usually includes cost and schedule estimates and justifies why the project contract should be awarded to a particular organization or team. Proposal writing is a critical task as the survival of many software companies depends on having enough proposals accepted and contracts awarded.

Project planning is an important topic in its own right, which I discuss in Chapter 23. In this chapter, I focus on risk management and people management.

22.1 Risk management

Risk management is one of the most important jobs for a project manager. You can think of a risk as something that you'd prefer not to have happen. Risks may threaten the project, the software that is being developed, or the organization. Risk management involves anticipating risks that might affect the project schedule or the quality of the software being developed, and then taking action to avoid these risks (Hall 1998; Ould 1999).

Risks can be categorized according to type of risk (technical, organizational, etc.), as I explain in Section 22.1.1. A complementary classification is to classify risks according to what these risks affect:

1. Project risks affect the project schedule or resources. An example of a project risk is the loss of an experienced system architect. Finding a replacement architect with appropriate skills and experience may take a long time; consequently, it will take longer to develop the software design than originally planned.
2. Product risks affect the quality or performance of the software being developed. An example of a product risk is the failure of a purchased component to perform

as expected. This may affect the overall performance of the system so that it is slower than expected.

3. Business risks affect the organization developing or procuring the software. For example, a competitor introducing a new product is a business risk. The introduction of a competitive product may mean that the assumptions made about sales of existing software products may be unduly optimistic.

Of course, these risk categories overlap. An experienced engineer's decision to leave a project, for example, presents a *project risk* because the software delivery schedule will be affected. It inevitably takes time for a new project member to understand the work that has been done, so he or she cannot be immediately productive. Consequently, the delivery of the system may be delayed. The loss of a team member can also be a *product risk* because a replacement may not be as experienced and so could make programming errors. Finally, losing a team member can be a *business risk* because an experienced engineer's reputation may be a critical factor in winning new contracts.

For large projects, you should record the results of the risk analysis in a risk register along with a consequence analysis. This sets out the consequences of the risk for the project, product, and business. Effective risk management makes it easier to cope with problems and to ensure that these do not lead to unacceptable budget or schedule slippage. For small projects, formal risk recording may not be required, but the project manager should be aware of them.

The specific risks that may affect a project depend on the project and the organizational environment in which the software is being developed. However, there are also common risks that are independent of the type of software being developed. These can occur in any software development project. Some examples of these common risks are shown in Figure 22.1.

Software risk management is important because of the inherent uncertainties in software development. These uncertainties stem from loosely defined requirements, requirements changes due to changes in customer needs, difficulties in estimating the time and resources required for software development, and differences in individual skills. You have to anticipate risks, understand their impact on the project, the product, and the business, and take steps to avoid these risks. You may need to draw up contingency plans so that, if the risks do occur, you can take immediate recovery action.

An outline of the process of risk management is presented in Figure 22.2. It involves several stages:

1. *Risk identification* You should identify possible project, product, and business risks.
2. *Risk analysis* You should assess the likelihood and consequences of these risks.
3. *Risk planning* You should make plans to address the risk, either by avoiding it or by minimizing its effects on the project.
4. *Risk monitoring* You should regularly assess the risk and your plans for risk mitigation and revise these plans when you learn more about the risk.

Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of company management with different priorities.
Hardware unavailability	Project	Hardware that 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.
Software tool underperformance	Product	Software tools that 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.

Figure 22.1 Examples of common project, product, and business risks

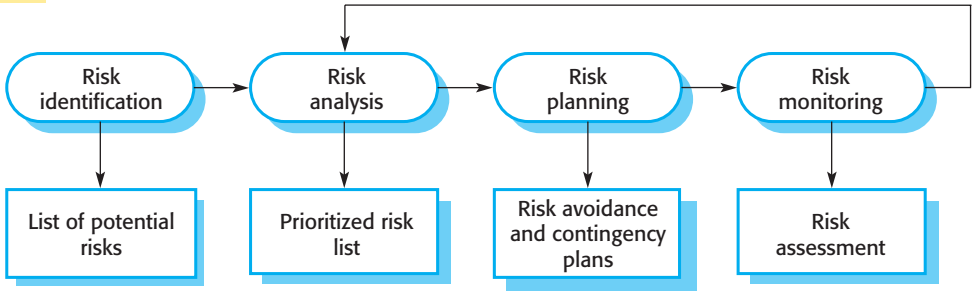

Figure 22.2 The risk management process

For large projects, you should document the outcomes of the risk management process in a risk management plan. This should include a discussion of the risks faced by the project, an analysis of these risks, and information on how you plan to manage the risk if it seems likely to be a problem.

The risk management process is an iterative process that continues throughout a project. Once you have drawn up an initial risk management plan, you monitor the situation to detect emerging risks. As more information about the risks becomes

available, you have to re-analyze the risks and decide if the risk priority has changed. You may then have to change your plans for risk avoidance and contingency management.

Risk management in agile development is less formal. The same fundamental activities should still be followed and risks discussed, although these may not be formally documented. Agile development reduces some risks, such as risks from requirements changes. However, agile development also has a downside. Because of its reliance on people, staff turnover can have significant effects on the project, product, and business. Because of the lack of formal documentation and its reliance on informal communications, it is very hard to maintain continuity and momentum if key people leave the project.

22.1.1 Risk identification

Risk identification is the first stage of the risk management process. It is concerned with identifying the risks that could pose a major threat to the software engineering process, the software being developed, or the development organization. Risk identification may be a team process in which a team gets together to brainstorm possible risks. Alternatively, project managers may identify risks based on their experience of what went wrong on previous projects.

As a starting point for risk identification, a checklist of different types of risk may be used. Six types of risk may be included in a risk checklist:

1. Estimation risks arise from the management estimates of the resources required to build the system.
2. Organizational risks arise from the organizational environment where the software is being developed.
3. People risks are associated with the people in the development team.
4. Requirements risks come from changes to the customer requirements and the process of managing the requirements change.
5. Technology risks come from the software or hardware technologies that are used to develop the system.
6. Tools risks come from the software tools and other support software used to develop the system.

Figure 22.3 shows examples of possible risks in each of these categories. When you have finished the risk identification process, you should have a long list of risks that could occur and that could affect the product, the process, and the business. You then need to prune this list to a manageable size. If you have too many risks, it is practically impossible to keep track of all of them.

Risk type	Possible risks
Estimation	1. The time required to develop the software is underestimated. 2. The rate of defect repair is underestimated. 3. The size of the software is underestimated.
Organizational	4. The organization is restructured so that different management are responsible for the project. 5. Organizational financial problems force reductions in the project budget.
People	6. It is impossible to recruit staff with the skills required. 7. Key staff are ill and unavailable at critical times. 8. Required training for staff is not available.
Requirements	9. Changes to requirements that require major design rework are proposed. 10. Customers fail to understand the impact of requirements changes.
Technology	11. The database used in the system cannot process as many transactions per second as expected. 12. Faults in reusable software components have to be repaired before these components are reused.
Tools	13. The code generated by software code generation tools is inefficient. 14. Software tools cannot work together in an integrated way.

Figure 22.3 Examples of different types of risk

22.1.2 Risk analysis

During the risk analysis process, you have to consider each identified risk and make a judgment about the probability and seriousness of that risk. There is no easy way to do so. You have to rely on your judgment and experience of previous projects and the problems that arose in them. It is not possible to make precise, numeric assessment of the probability and seriousness of each risk. Rather, you should assign the risk to one of a number of bands:

1. The probability of the risk might be assessed as insignificant, low, moderate, high, or very high.
2. The effects of the risk might be assessed as catastrophic (threaten the survival of the project), serious (would cause major delays), tolerable (delays are within allowed contingency), or insignificant.

You may then tabulate the results of this analysis process using a table ordered according to the seriousness of the risk. Figure 22.4 illustrates this for the risks that I have identified in Figure 22.3. Obviously, the assessment of probability and seriousness is arbitrary here. To make this assessment, you need

Risk	Probability	Effects
Organizational financial problems force reductions in the project budget (5).	Low	Catastrophic
It is impossible to recruit staff with the skills required (6).	High	Catastrophic
Key staff are ill at critical times in the project (7).	Moderate	Serious
Faults in reusable software components have to be repaired before these components are reused (12).	Moderate	Serious
Changes to requirements that require major design rework are proposed (9).	Moderate	Serious
The organization is restructured so that different managements are responsible for the project (4).	High	Serious
The database used in the system cannot process as many transactions per second as expected (11).	Moderate	Serious
The time required to develop the software is underestimated (1).	High	Serious
Software tools cannot be integrated (14).	High	Tolerable
Customers fail to understand the impact of requirements changes (10).	Moderate	Tolerable
Required training for staff is not available (8).	Moderate	Tolerable
The rate of defect repair is underestimated (2).	Moderate	Tolerable
The size of the software is underestimated (3).	High	Tolerable
Code generated by code generation tools is inefficient (13).	Moderate	Insignificant

Figure 22.4 Risk types and examples

detailed information about the project, the process, the development team, and the organization.

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. You should therefore update this table during each iteration of the risk management process.

Once the risks have been analyzed and ranked, you should assess which of these risks are most significant. Your judgment must depend on a combination of the probability of the risk arising and the effects of that risk. In general, catastrophic risks should always be considered, as should all serious risks that have more than a moderate probability of occurrence.

Boehm (Boehm 1988) recommends identifying and monitoring the "top 10" risks. However, I think that the right number of risks to monitor must depend on the project. It might be 5 or it might be 15. From the risks identified in Figure 22.4, I think that it is appropriate to consider the eight risks that have catastrophic or serious consequences (Figure 22.5).

22.1.3 Risk planning

The risk planning process develops strategies to manage the key risks that threaten the project. For each risk, you have to think of actions that you might take to minimize the disruption to the project if the problem identified in the risk occurs. You should also think about the information that you need to collect while monitoring the project so that emerging problems can be detected before they become serious.

In risk planning, you have to ask "what-if" questions that consider both individual risks, combinations of risks, and external factors that affect these risks. For example, questions that you might ask are:

1. What if several engineers are ill at the same time?
2. What if an economic downturn leads to budget cuts of 20% for the project?
3. What if the performance of open-source software is inadequate and the only expert on that open-source software leaves?
4. What if the company that supplies and maintains software components goes out of business?
5. What if the customer fails to deliver the revised requirements as predicted?

Based on the answers to these "what-if" questions, you may devise strategies for managing the risks. Figure 22.5 shows possible risk management strategies that have been identified for the key risks (i.e., those that are serious or intolerable) shown in Figure 22.4. These strategies fall into three categories:

1. *Avoidance strategies* Following these strategies means that the probability that the risk will arise is reduced. An example of a risk avoidance strategy is the strategy for dealing with defective components shown in Figure 22.5.
2. *Minimization strategies* Following these strategies means that the impact of the risk is reduced. An example of a risk minimization strategy is the strategy for staff illness shown in Figure 22.5.
3. *Contingency plans* Following these strategies means that you are prepared for the worst and have a strategy in place to deal with it. An example of a contingency strategy is the strategy for organizational financial problems that I have shown in Figure 22.5.

You can see a clear analogy here with the strategies used in critical systems to ensure reliability, security, and safety, where you must avoid, tolerate, or recover from failures. Obviously, it is best to use a strategy that avoids the risk. If this is not possible, you should use a strategy that reduces the chances that the risk will have serious effects. Finally, you should have strategies in place to

Risk	Strategy
Organizational 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 and presenting reasons why cuts to the project budget would not be cost-effective.
Recruitment problems	Alert customer to potential difficulties and the possibility of delays; investigate buying-in components.
Staff illness	Reorganize 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; maximize information hiding in the design.
Organizational 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 use of automated code generation.

Figure 22.5 Strategies to help manage risk

cope with the risk if it arises. These should reduce the overall impact of a risk on the project or product.

22.1.4 Risk monitoring

Risk monitoring is the process of checking that your assumptions about the product, process, and business risks have not changed. You should regularly assess each of the identified risks to decide whether or not that risk is becoming more or less probable. You should also think about whether or not the effects of the risk have changed. To do this, you have to look at other factors, such as the number of requirements change requests, which give you clues about the risk probability and its effects. These factors are obviously dependent on the types of risk. Figure 22.6 gives some examples of factors that may be helpful in assessing these risk types.

You should monitor risks regularly at all stages in a project. At every management review, you should consider and discuss each of the key risks separately. You should decide if the risk is more or less likely to arise and if the seriousness and consequences of the risk have changed.

Risk type	Potential indicators
Estimation	Failure to meet agreed schedule; failure to clear reported defects.
Organizational	Organizational gossip; lack of action by senior management.
People	Poor staff morale; poor relationships among team members; high staff turnover.
Requirements	Many requirements change requests; customer complaints.
Technology	Late delivery of hardware or support software; many reported technology problems.
Tools	Reluctance by team members to use tools; complaints about software tools; requests for faster computers/more memory, and so on.

Figure 22.6 Risk indicators

22.2 Managing people

The people working in a software organization are its greatest assets. It is expensive to recruit and retain good people, and it is up to software managers to ensure that the engineers working on a project are as productive as possible. In successful companies and economies, this productivity is achieved when people are respected by the organization and are assigned responsibilities that reflect their skills and experience.

It is important that software project managers understand the technical issues that influence the work of software development. Unfortunately, however, good software engineers are not always good people managers. Software engineers often have strong technical skills but may lack the softer skills that enable them to motivate and lead a project development team. As a project manager, you should be aware of the potential problems of people management and should try to develop people management skills.

There are four critical factors that influence the relationship between a manager and the people that he or she manages:

1. *Consistency* All the people in a project team should be treated in a comparable way. No one expects all rewards to be identical, but people should not feel that their contribution to the organization is undervalued.
2. *Respect* Different people have different skills, and managers should respect these differences. All members of the team should be given an opportunity to make a contribution. In some cases, of course, you will find that people simply don't fit into a team and they cannot continue, but it is important not to jump to conclusions about them at an early stage in the project.

3. *Inclusion* People contribute effectively when they feel that others listen to them and take account of their proposals. It is important to develop a working environment where all views, even those of the least experienced staff, are considered.

4. *Honesty* As a manager, you should always be honest about what is going well and what is going badly in the team. You should also be honest about your level of technical knowledge and be willing to defer to staff with more knowledge when necessary. If you try to cover up ignorance or problems, you will eventually be found out and will lose the respect of the group.

Practical people management has to be based on experiences so my aim in this section and the following section on teamwork is to raise awareness of the most important issues that project managers may have to deal with.

22.2.1 Motivating people

As a project manager, you need to motivate the people who work with you so that they will contribute to the best of their abilities. In practice, motivation means organizing work and its environment to encourage people to work as effectively as possible. If people are not motivated, they will be less interested in the work they are doing. They will work slowly, be more likely to make mistakes, and will not contribute to the broader goals of the team or the organization.

To provide this encouragement, you should understand a little about what motivates people. Maslow (Maslow 1954) suggests that people are motivated by satisfying their needs. These needs are arranged in a series of levels, as shown in Figure 22.7. The lower levels of this hierarchy represent fundamental needs for food, sleep, and so on, and the need to feel secure in an environment. Social need is concerned with the need to feel part of a social grouping. Esteem need represents the need to feel respected by others, and self-realization need is concerned with personal development. People need to satisfy lower-level needs such as hunger before the more abstract, higher-level needs.

People working in software development organizations are not usually hungry, thirsty, or physically threatened by their environment. Therefore, making sure that peoples' social, esteem, and self-realization needs are satisfied is most important from a management point of view.

1. To satisfy social needs, you need to give people time to meet their co-workers and provide places for them to meet. Software companies such as Google provide social space in their offices for people to get together. This is relatively easy when all of the members of a development team work in the same place, but, increasingly, team members are not located in the same building or even the same town or state. They may work for different organizations or from home most of the time.

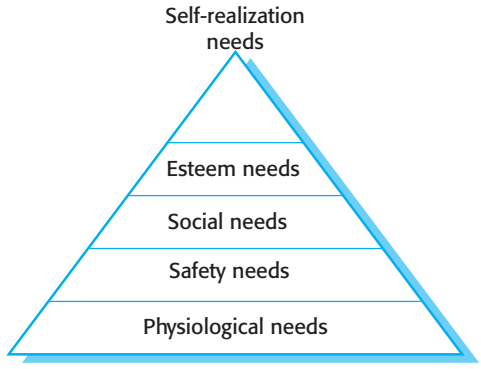

Figure 22.7 Human needs hierarchy

Social networking systems and teleconferencing can be used for remote communications, but my experience with these systems is that they are most effective when people already know each other. You should arrange some face-to-face meetings early in the project so that people can directly interact with other members of the team. Through this direct interaction, people become part of a social group and accept the goals and priorities of that group.

2. To satisfy esteem needs, you need to show people that they are valued by the organization. Public recognition of achievements is a simple and effective way of doing this. Obviously, people must also feel that they are paid at a level that reflects their skills and experience.

3. Finally, to satisfy self-realization needs, you need to give people responsibility for their work, assign them demanding (but not impossible) tasks, and provide opportunities for training and development where people can enhance their skills. Training is an important motivating influence as people like to gain new knowledge and learn new skills.

Maslow's model of motivation is helpful up to a point, but I think that a problem with it is that it takes an exclusively personal viewpoint on motivation. It does not take adequate account of the fact that people feel themselves to be part of an organization, a professional group, and one or more cultures. Being a member of a cohesive group is highly motivating for most people. People with fulfilling jobs often like to go to work because they are motivated by the people they work with and the work that they do. Therefore, as a manager, you also have to think about how a group as a whole can be motivated. I discuss this and other teamwork issues in Section 22.3.

In Figure 22.8, I illustrate a problem of motivation that managers often have to face. In this example, a competent group member loses interest in the work and in the group as a whole. The quality of her work falls and becomes unacceptable. This situation has to be dealt with quickly. If you don't sort out the problem, the other group members will become dissatisfied and feel that they are doing an unfair share of the work.

Case study: Motivation

Alice is a software project manager working in a company that develops alarm systems. This company wishes to enter the growing market of assistive technology to help elderly and disabled people live independently. Alice has been asked to lead a team of six developers that can develop new products based on the company's alarm technology.

Alice's assistive technology project starts well. Good working relationships develop within the team, and creative new ideas are developed. The team decides to develop a system that a user can initiate and control the alarm system from a cell phone or tablet computer. However, some months into the project, Alice notices that Dorothy, a hardware expert, starts coming into work late, that the quality of her work is deteriorating, and, increasingly, that she does not appear to be communicating with other members of the team.

Alice talks about the problem informally with other team members to try to find out if Dorothy's personal circumstances have changed and if this might be affecting her work. They don't know of anything, so Alice decides to talk with Dorothy to try to understand the problem.

After some initial denials of any problem, Dorothy admits that she has lost interest in the job. She expected that she would be able to develop and use her hardware interfacing skills. However, because of the product direction that has been chosen, she has little opportunity to use these skills. Basically, she is working as a C programmer on the alarm system software.

While she admits that the work is challenging, she is concerned that she is not developing her interfacing skills. She is worried that finding a job that involves hardware interfacing will be difficult after this project. Because she does not want to upset the team by revealing that she is thinking about the next project, she has decided that it is best to minimize conversation with them.

Figure 22.8 Individual motivation

In this example, Alice tries to find out if Dorothy's personal circumstances could be the problem. Personal difficulties commonly affect motivation because people cannot therefore concentrate on their work. You may have to give them time and support to resolve these issues, although you also have to make it clear that they still have a responsibility to their employer.

Dorothy's motivation problem is one that can arise when projects develop in an unexpected direction. People who expect to do one type of work may end up doing something completely different. In those circumstances, you may decide that the team member should leave the team and find opportunities elsewhere. In this example, however, Alice decides to try to convince Dorothy that broadening her experience is a positive career step. She gives Dorothy more design autonomy and organizes training courses in software engineering that will give her more opportunities after her current project has finished.

Psychological personality type also influences motivation. Bass and Dunteman (Bass and Dunteman 1963) identified three classifications for professional workers:

1. *Task-oriented people,* who are motivated by the work they do. In software engineering, these are people who are motivated by the intellectual challenge of software development.

The People Capability Maturity Model

The People Capability Maturity Model (P-CMM) is a framework for assessing how well organizations manage the development of their staff. It highlights best practice in people management and provides a basis for organizations to improve their people management processes. It is best suited to large rather than small, informal companies.

http://software-engineering-book.com/web/people-cmm/

2. *Self-oriented people,* who are principally motivated by personal success and recognition. They are interested in software development as a means of achieving their own goals. They often have longer-term goals, such as career progression, that motivate them, and they wish to be successful in their work to help realize these goals.

3. *Interaction-oriented people,* who are motivated by the presence and actions of co-workers. As more and more attention is paid to user interface design, interaction-oriented individuals are becoming more involved in software engineering.

Research has shown that interaction-oriented personalities usually like to work as part of a group, whereas task-oriented and self-oriented people usually prefer to act as individuals. Women are more likely to be interaction-oriented than men are. They are often more effective communicators. I discuss the mix of these different personality types in groups in the case study shown later in Figure 22.10.

Each individual's motivation is made up of elements of each class, but one type of motivation is usually dominant at any one time. However, individuals can change. For example, technical people who feel they are not being properly rewarded can become self-oriented and put personal interests before technical concerns. If a group works particularly well, self-oriented people can become more interaction-oriented.

22.3 Teamwork

Most professional software is developed by project teams that range in size from two to several hundred people. However, as it is impossible for everyone in a large group to work together on a single problem, large teams are usually split into a number of smaller groups. Each group is responsible for developing part of the overall system. The best size for a software engineering group is 4 to 6 members, and they should never have more than 12 members. When groups are small, communication problems are reduced. Everyone knows everyone else, and the whole group can get around a table for a meeting to discuss the project and the software that they are developing.

Putting together a group that has the right balance of technical skills, experience, and personalities is a critical management task. However, successful groups are more than simply a collection of individuals with the right balance of skills. A good group is cohesive and thinks of itself as a strong, single unit. The people involved are motivated by the success of the group as well as by their own personal goals.

In a cohesive group, members think of the group as more important than the individuals who are group members. Members of a well-led, cohesive group are loyal to the group. They identify with group goals and other group members. They attempt to protect the group, as an entity, from outside interference. This makes the group robust and able to cope with problems and unexpected situations.

The benefits of creating a cohesive group are:

1. *The group can establish its own quality standards* Because these standards are established by consensus, they are more likely to be observed than external standards imposed on the group.
2. *Individuals learn from and support each other* Group members learn by working together. Inhibitions caused by ignorance are minimized as mutual learning is encouraged.
3. *Knowledge is shared* Continuity can be maintained if a group member leaves. Others in the group can take over critical tasks and ensure that the project is not unduly disrupted.
4. *Refactoring and continual improvement is encouraged* Group members work collectively to deliver high-quality results and fix problems, irrespective of the individuals who originally created the design or program.

Good project managers should always try to encourage group cohesiveness. They may try to establish a sense of group identity by naming the group and establishing a group identity and territory. Some managers like explicit group-building activities such as sports and games, although these are not always popular with group members. Social events for group members and their families are a good way to bring people together.

One of the most effective ways of promoting cohesion is to be inclusive. That is, you should treat group members as responsible and trustworthy, and make information freely available. Sometimes managers feel that they cannot reveal certain information to everyone in the group. This invariably creates a climate of mistrust. An effective way of making people feel valued and part of a group is to make sure that they know what is going on.

You can see an example in the case study in Figure 22.9. Alice arranges regular informal meetings where she tells the other group members what is going on. She makes a point of involving people in the product development by asking them to come up with new ideas derived from their own family experiences. The "away

Case study: Team spirit

Alice, an experienced project manager, understands the importance of creating a cohesive group. As her company is developing a new product, she takes the opportunity to involve all group members in the product specification and design by getting them to discuss possible technology with elderly members of their families. She encourages them to bring these family members to meet other members of the development group.

Alice also arranges monthly lunches for everyone in the group. These lunches are an opportunity for all team members to meet informally, talk around issues of concern, and get to know each other. At the lunch, Alice tells the group what she knows about organizational news, policies, strategies, and so forth. Each team member then briefly summarizes what they have been doing, and the group discusses a general topic, such as new product ideas from elderly relatives.

Every few months, Alice organizes an "away day" for the group where the team spends two days on "technology updating." Each team member prepares an update on a relevant technology and presents it to the group. This is an offsite meeting, and plenty of time is scheduled for discussion and social interaction.

Figure 22.9 Group cohesion

days" are also good ways of promoting cohesion: People relax together while they help each other learn about new technologies.

Whether or not a group is effective depends, to some extent, on the nature of the project and the organization doing the work. If an organization is in a state of turmoil with constant reorganizations and job insecurity, it is difficult for team members to focus on software development. Similarly, if a project keeps changing and is in danger of cancellation, people lose interest in it.

Given a stable organizational and project environment, the three factors that have the biggest effect on team working are:

1. *The people in the group* You need a mix of people in a project group as software development involves diverse activities such as negotiating with clients, programming, testing, and documentation.
2. *The way the group is organized* A group should be organized so that individuals can contribute to the best of their abilities and tasks can be completed as expected.
3. *Technical and managerial communications* Good communication between group members, and between the software engineering team and other project stakeholders, is essential.

As with all management issues, getting the right team cannot guarantee project success. Too many other things can go wrong, including changes to the business and the business environment. However, if you don't pay attention to group composition, organization, and communications, you increase the likelihood that your project will run into difficulties.

22.3.1 Selecting group members

A manager or team leader's job is to create a cohesive group and organize that group so that they work together effectively. This task involves selecting a group with the right balance of technical skills and personalities. Sometimes people are hired from outside the organization; more often, software engineering groups are put together from current employees who have experience on other projects. Managers rarely have a completely free hand in team selection. They often have to use the people who are available in the company, even if they are not the ideal people for the job.

Many software engineers are motivated primarily by their work. Software development groups, therefore, are often composed of people who have their own ideas about how technical problems should be solved. They want to do the best job possible, so they may deliberately redesign systems that they think can be improved and add extra system features that are not in the system requirements. Agile methods encourage engineers to take the initiative to improve the software. However, sometimes this means that time is spent doing things that aren't really needed and that different engineers compete to rewrite each other's code.

Technical knowledge and ability should not be the only factor used to select group members. The "competing engineers" problem can be reduced if the people in the group have complementary motivations. People who are motivated by the work are likely to be the strongest technically. People who are self-oriented will probably be best at pushing the work forward to finish the job. People who are interaction-oriented help facilitate communications within the group. I think that it is particularly important to have interaction-oriented people in a group. They like to talk to people and can detect tensions and disagreements at an early stage, before these problems have a serious impact on the group.

In the case study in Figure 22.10, I have suggested how Alice, the project manager, has tried to create a group with complementary personalities. This particular group has a good mix of interaction- and task-oriented people, but I have already discussed, in Figure 22.8, how Dorothy's self-oriented personality has caused problems because she has not been doing the work that she expected. Fred's part-time role in the group as a domain expert might also be a problem. He is mostly interested in technical challenges, so he may not interact well with other group members. The fact that he is not always part of the team means that he may not fully relate to the team's goals.

It is sometimes impossible to choose a group with complementary personalities. If this is the case, the project manager has to control the group so that individual goals do not take precedence over organizational and group objectives. This control is easier to achieve if all group members participate in each stage of the project. Individual initiative is most likely to develop when group members are given instructions without being aware of the part that their task plays in the overall project.

For example, say a software engineer takes over the development of a system and notices that possible improvements could be made to the design. If he or she implements these improvements without understanding the rationale for the original design, any changes, though well-intentioned, might have adverse implications for

Case study: Group composition

In creating a group for assistive technology development, Alice is aware of the importance of selecting members with complementary personalities. When interviewing potential group members, she tried to assess whether they were task-oriented, self-oriented, or interaction-oriented. She felt that she was primarily a self-oriented type because she considered the project to be a way of getting noticed by senior management and possibly being promoted. She therefore looked for one or perhaps two interaction-oriented personalities, with task-oriented individuals to complete the team. The final assessment that she arrived at was:

Alice—self-oriented
Brian—task-oriented
Chun—interaction-oriented
Dorothy—self-oriented
Ed—interaction-oriented
Fiona—task-oriented
Fred—task-oriented
Hassan—interaction-oriented

Figure 22.10 Group composition

other parts of the system. If all the members of the group are involved in the design from the start, they are more likely to understand why design decisions have been made. They may then identify with these decisions rather than oppose them.

22.2.3 Group organization

The way a group is organized affects the group's decisions, the ways information is exchanged, and the interactions between the development group and external project stakeholders. Important organizational questions for project managers include the following:

1. Should the project manager be the technical leader of the group? The technical leader or system architect is responsible for the critical technical decisions made during software development. Sometimes the project manager has the skill and experience to take on this role. However, for large projects, it is best to separate technical and managerial roles. The project manager should appoint a senior engineer to be the project architect, who will take responsibility for technical leadership.
2. Who will be involved in making critical technical decisions, and how will these decisions be made? Will decisions be made by the system architect or the project manager or by reaching consensus among a wider range of team members?
3. How will interactions with external stakeholders and senior company management be handled? In many cases, the project manager will be responsible for these interactions, assisted by the system architect if there is one. However, an alternative organizational model is to create a dedicated role concerned with external liaison and appoint someone with appropriate interaction skills to that role.

Hiring the right people

Project managers are often responsible for selecting the people in the organization who will join their software engineering team. Getting the best possible people in this process is very important as poor selection decisions may be a serious risk to the project.

Key factors that should influence the selection of staff are education and training, application domain and technology experience, communication ability, adaptability, and problem solving ability.

http://software-engineering-book.com/web/people-selection/

4. How can groups integrate people who are not co-located? It is now common for groups to include members from different organizations and for people to work from home as well as in a shared office. This change has to be considered in group decision-making processes.

5. How can knowledge be shared across the group? Group organization affects information sharing as certain methods of organization are better for sharing than others. However, you should avoid too much information sharing as people become overloaded and excessive information distracts them from their work.

Small programming groups are usually organized in an informal way. The group leader gets involved in the software development with the other group members. In an informal group, the group as a whole discusses the work to be carried out, and tasks are allocated according to ability and experience. More senior group members may be responsible for the architectural design. However, detailed design and implementation is the responsibility of the team member who is allocated to a particular task.

Agile development teams are always informal groups. Agile enthusiasts claim that formal structure inhibits information exchange. Many decisions that are usually seen as management decisions (such as decisions on schedule) may be devolved to group members. However, there still needs to be a project manager who is responsible for strategic decision making and communications outside of the group.

Informal groups can be very successful, particularly when most group members are experienced and competent. Such a group makes decisions by consensus, which improves cohesiveness and performance. However, if a group is composed mostly of inexperienced or incompetent members, informality can be a hindrance. With no experienced engineers to direct the work, the result can be a lack of coordination between group members and, possibly, eventual project failure.

In hierarchical groups the group leader is at the top of the hierarchy. He or she has more formal authority than the group members and so can direct their work. There is a clear organizational structure, and decisions are made toward the top of the hierarchy and implemented by people lower down. Communications are primarily instructions from senior staff; the people at lower levels of the hierarchy have relatively little communication with the managers at the upper levels.

Hierarchical groups can work well when a well-understood problem can be easily broken down into software components that can be developed in different parts of the hierarchy. This grouping allows for rapid decision making, which is why military organizations follow this model. However, it rarely works well for complex software engineering. In software development, effective team communications at all levels is essential:

1. Changes to the software often require changes to several parts of the system, and this requires discussion and negotiation at all levels in the hierarchy.
2. Software technologies change so fast that more junior staff may know more about new technologies than experienced staff. Top-down communications may mean that the project manager does not find out about the opportunities of using these new technologies. More junior staff may become frustrated because of what they see as old-fashioned technologies being used for development.

A major challenge facing project managers is the difference in technical ability between group members. The best programmers may be up to 25 times more productive than the worst programmers. It makes sense to use these "super-programmers" in the most effective way and to provide them with as much support as possible.

At the same time, focusing on the super-programmers can be demotivating for other group members who are resentful that they are not given responsibility. They may be concerned that this will affect their career development. Furthermore, if a "super-programmer" leaves the company, the impact on a project can be huge. Therefore, adopting a group model that is based on individual experts can pose significant risks.

22.3.3 Group communications

It is absolutely essential that group members communicate effectively and efficiently with each other and with other project stakeholders. Group members must exchange information on the status of their work, the design decisions that have been made, and changes to previous design decisions. They have to resolve problems that arise with other stakeholders and inform these stakeholders of changes to the system, the group, and delivery plans. Good communication also helps strengthen group cohesiveness. Group members come to understand the motivations, strengths, and weaknesses of other people in the group.

The effectiveness and efficiency of communications are influenced by:

1. *Group size* As a group gets bigger, it gets harder for members to communicate effectively. The number of one-way communication links is $n * (n - 1)$, where n is the group size, so, with a group of eight members, there are 56 possible communication pathways. This means that it is quite possible that some people will rarely communicate with each other. Status differences between group members mean that communications are often one-way. Managers and experienced engineers tend to dominate communications with less experienced staff, who may be reluctant to start a conversation or make critical remarks.

The physical work environment

Group communications and individual productivity are both affected by the team's working environment. Individual workspaces are better for concentration on detailed technical work as people are less likely to be distracted by interruptions. However, shared workspaces are better for communications. A well-designed work environment takes both of these needs into account.

http://software-engineering-book.com/web/workspace/

2. *Group structure* People in informally structured groups communicate more effectively than people in groups with a formal, hierarchical structure. In hierarchical groups, communications tend to flow up and down the hierarchy. People at the same level may not talk to each other. This is a particular problem in a large project with several development groups. If people working on different subsystems only communicate through their managers, then there are more likely to be delays and misunderstandings.

3. *Group composition* People with the same personality types (discussed in Section 22.2) may clash, and, as a result, communications can be inhibited. Communication is also usually better in mixed-sex groups than in single-sex groups (Marshall and Heslin 1975). Women are often more interaction-oriented than men and may act as interaction controllers and facilitators for the group.

4. *The physical work environment* The organization of the workplace is a major factor in facilitating or inhibiting communications. While some companies use standard open-plan offices for their staff, others invest in providing a workspace that includes a mixture of private and group working areas. This allows for both collaborative activities and individual development that require a high level of concentration.

5. *The available communication channels* There are many different forms of communication—face to face, email messages, formal documents, telephone, and technologies such as social networking and wikis. As project teams become increasingly distributed, with team members working remotely, you need to make use of interaction technologies, such as conferencing systems, to facilitate group communications.

Project managers usually work to tight deadlines, and, consequently, they often try to use communication channels that don't take up too much of their time. They may rely on meetings and formal documents to pass on information to project staff and stakeholders and send long emails to project staff. Unfortunately, while this may be an efficient approach to communication from a project manager's perspective, it is not usually very effective. There are often good reasons why people can't attend meetings, and so they don't hear the presentation. People do not have time to read long documents and emails that are not directly relevant to their work. When several versions of the same document are produced, readers find it difficult to keep track of the changes.

Effective communication is achieved when communications are two-way and the people involved can discuss issues and information and establish a common understanding of proposals and problems. All this can be done through meetings, although these meetings are often dominated by powerful personalities. Informal discussions when a manager meets with the team for coffee are sometimes more effective.

More and more project teams include remote members, which also makes meetings more difficult. To involve them in communications, you may make use of wikis and blogs to support information exchange. Wikis support the collaborative creation and editing of documents, and blogs support threaded discussions about questions and comments made by group members. Wikis and blogs allow project members and external stakeholders to exchange information, irrespective of their location. They help manage information and keep track of discussion threads, which often become confusing when conducted by email. You can also use instant messaging and teleconferences, which can be easily arranged, to resolve issues that need discussion.

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 as mature as traditional engineering processes.
- Risk management involves identifying and assessing major project risks to establish the probability that they will occur and the consequences for the project if that risk does arise. You should make plans to avoid, manage, or deal with likely risks if or when they arise.
- People management involves choosing the right people to work on a project and organizing the team and its working environment so that they are as productive as possible.
- People are motivated by interaction with other people, by the recognition of management and their peers, and by being given opportunities for personal development.
- Software development groups should be fairly small and cohesive. The key factors that influence the effectiveness of a group are the people in that group, the way that it is organized, and the communication between group members.
- Communications within a group are influenced by factors such as the status of group members, the size of the group, the gender composition of the group, personalities, and available communication channels.

FURTHER READING

The Mythical Man Month: Essays on Software Engineering (Anniversary Edition). The problems of software management have remained largely unchanged since the 1960s, and this is one of the best books on the topic. It presents an interesting and readable account of the management of one of the first very large software projects, the IBM OS/360 operating system. The anniversary edition (published 20 years after the original edition in 1975) includes other classic papers by Brooks. (F. P. Brooks, 1995, Addison-Wesley).

Peopleware: Productive Projects and Teams, 2nd ed. This now classic book focuses on the importance of treating people properly when managing software projects. It is one of the few books that recognizes how the place where people work influences communications and productivity. Strongly recommended. (T. DeMarco and T. Lister, 1999, Dorset House).

Waltzing with Bears: Managing Risk on Software Projects. A very practical and easy-to-read introduction to risks and risk management. (T. DeMarco and T. Lister, 2003, Dorset House).

Effective Project Management: Traditional, Agile, Extreme. 2014 (7th ed.). This is a textbook on project management in general rather than software project management. It is based on the so-called PMBOK (Project Management Body of Knowledge) and, unlike most books on this topic, discusses PM techniques for agile projects. (R. K. Wysocki, 2014).

WEBSITE

PowerPoint slides for this chapter:

www.pearsonglobaleditions.com/Sommerville

Links to supporting videos:

http://software-engineering-book.com/videos/software-management/

EXERCISES

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

22.2. Explain how company size and software size are factors that affect software project management.

22.3. Using reported instances of project problems in the literature, list management difficulties and errors that occurred in these failed programming projects. (I suggest that you start with *The Mythical Man Month*, as suggested in Further Reading.)

22.4. In addition to the risks shown in Figure 22.1, identify at least six other possible risks that could arise in software projects.

22.5. What is risk monitoring? How can risks be monitored? List a few examples of types of risks and their potential indicators.

22.6. Fixed-price contracts, where the contractor bids a fixed price to complete a system development, may be used to move project risk from client to contractor. If anything goes wrong, the contractor has to pay. Suggest how the use of such contracts may increase the likelihood that product risks will arise.

22.7. Explain why keeping all members of a group informed about progress and technical decisions in a project can improve group cohesiveness.

22.8. What qualities of a cohesive group's members make the group robust? List out the key benefits of creating a cohesive group.

22.9. Write a case study in the style used here to illustrate the importance of communications in a project team. Assume that some team members work remotely and that it is not possible to get the whole team together at short notice.

22.10. Your manager asks you to deliver software to a schedule that 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 organization rather than to their families. What factors might be significant in your decision?

REFERENCES

Bass, B. M., and G. Dunteman. 1963. "Behaviour in Groups as a Function of Self, Interaction and Task Orientation." *J. Abnorm. Soc. Psychology*. 66 (4): 19–28. doi:10.1037/h0042764.

Boehm, B. W. 1988. "A Spiral Model of Software Development and Enhancement." *IEEE Computer* 21 (5): 61–72. doi:10.1109/2.59.

Hall, E. 1998. *Managing Risk: Methods for Software Systems Development*. Reading, MA: Addison-Wesley.

Marshall, J. E., and R. Heslin. 1975. "Boys and Girls Together. Sexual Composition and the Effect of Density on Group Size and Cohesiveness." *J. of Personality and Social Psychology* 35 (5): 952–961. doi:10.1037/h0076838.

Maslow, A. A. 1954. *Motivation and Personality*. New York: Harper & Row.

Ould, M. 1999. *Managing Software Quality and Business Risk*. Chichester, UK: John Wiley & Sons.