12 Working in Teams

Learning Objectives

- Improve group working
- Analyse the coordination needs of a project
- Select the best communication genres to support the coordination needs of a project
- Draw up a communication plan
- Evaluate the characteristics of the various team structures
- Use the most appropriate leadership styles

12.1 Introduction

We associate software development with advanced technologies yet it is a task requiring intense human mental activity. Software-based systems can be huge – the software to control a telephone switching system can contain five million lines of code – so that this human effort has to be shared between individual software developers within teams and between groups of developers. Amanda at IOE wants to get the best out of her team, but also needs to coordinate the work of her group with other parts of IOE, including the users. At Brightmouth College, Brigette does not have a big team to manage, but the need to coordinate her efforts with those of other project stakeholders is probably greater.

This chapter will look enhancing communication between individual developers within teams and across teams. It will also look at how the efforts of individuals and teams can be coordinated through communication.

By 'teams' we usually mean groups of people who are working together. Typically the individuals work in the same office, that is, are *co-located* – although we will see that this is not always so. However, the term 'project team' is sometimes used to refer to *all* the people working on a project. These people may sit in different work groups at some distance from each other. These groups can also change over time. Thus individual software developers are likely to transfer between teams as projects start and finish.

We will start by looking at the small group environment where the term 'team' is perhaps most justified. We will look at how true teams come to be formed. We will see how, apart from their technical roles, team members take on social roles that help team effectiveness.

A team is created to carry out a joint assignment. We will see how some tasks contributing to project objectives are best done by an individual. Other tasks, usually those that involve judgement or decision-making, may be better done by groups.

We will look at how teams can be coordinated. An organization needs to control the allocation of staff to work assignments. This is one form of coordination needed between groups and individuals within a project and other types will be outlined.

Communication genres refer to methods of communication. This goes beyond technologies used and includes the organizational conventions involved in the communication. Communication genres can be selected and developed to deal with particular need for project coordination. We will see how arrangements for communication between project stakeholders can be documented in a *communication plan*.

As well as coordination which reacts to day-to-day problems, but there needs to be proactive central direction. This introduces issues related to leadership.

The collaborative nature of project work will have an influence on nearly all stages of the Step Wise project planning framework (Figure 12.1).

1. *Identify project scope and objectives*. Here stakeholders in the project are identified and communications channels are established.
2. *Identify project infrastructure*. The organization structure within which the project team will exist is identified.
3. *Analyse project characteristics*. Decisions made about how the project is to be executed – for example buying versus building software functionality – will affect the team structure needed.
4. *Estimate effort for each activity*. Individual and group experience will have a key influence on developer productivity.
5. *Identify activity risks*. Risks will include those that relate to staff such as continued availability.
6. *Allocate resources*.
7. *Review/publicize plan*. A communication plan could be produced at this point.

12.2 Becoming a Team

First we look at how small work groups – where the description 'team' is perhaps most apt – are formed. Simply throwing people together will not immediately enable them to work together as a team. It is suggested that teams go through five basic stages of development:

- *Forming* The members of the group get to know each other and try to set up some ground rules about behaviour.
- *Storming* Conflicts arise as various members of the group try to exert leadership and the group's methods of operation are being established.
- *Norming* Conflicts are largely settled and a feeling of group identity emerges.
- *Performing* The emphasis is now on the tasks at hand.
- *Adjourning* The group disbands.

This classification is associated with B. W. Tuckman and M. A. Jensen.

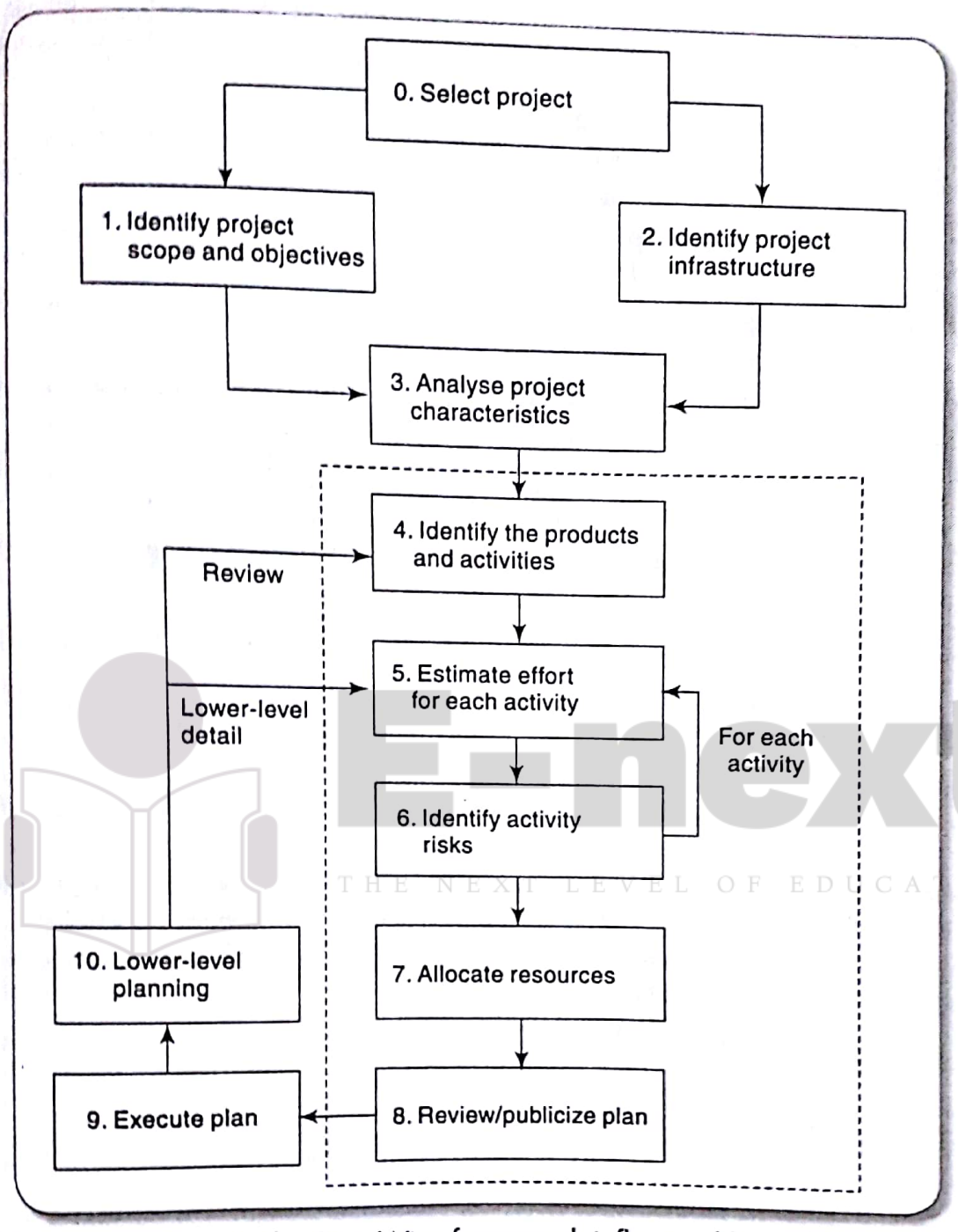

FIGURE 12.1 Some places in the Step Wise framework influenced by collaborative working

Sometimes specific team-building exercises can be undertaken. Some organizations, for example, send their management teams off on outdoor activities. Without going to these lengths, Amanda and Brigette might devise some training activities which promote team building.

Valuable research has examined the best mix of personalities in a project team. Belbin studied teams working together on management games. He initially tried putting the most able people into one group. Surprisingly, these élite teams tended to do very badly – they argued a lot and as a result important tasks were often neglected.

Belbin came to the conclusion that teams needed a balance of different types of people.

R. Meredith Belbin (2003) *Management Teams: Why They Succeed or Fail*, 2nd edition, Elsevier, contains a self-assessment questionnaire which identifies the role a person is best suited to.

- *The chair:* not necessarily brilliant leaders but they must be good at running meetings, being calm, strong but tolerant.
- *The plant:* someone who is essentially very good at generating ideas and potential solutions to problems.
- *The monitor–evaluator:* good at evaluating ideas and potential solutions and helping to select the best one.
- *The shaper:* rather a worrier, who helps to direct the team's attention to the important issues.
- *The team worker:* skilled at creating a good working environment, for example, by 'jollying people along'.
- *The resource investigator:* adept at finding resources in terms of both physical resources and information.
- *The completer–finisher:* concerned with completing tasks.
- *The company worker:* a good team player who is willing to undertake less attractive tasks if they are needed for team success.

In *Team roles at work*, 1996, Belbin suggests that 'coordinator' and 'implementer' are better descriptions than 'chair' and 'team worker'. A new role is added: the 'specialist', the 'techie' who likes to acquire knowledge for its own sake.

A person can have elements of more than one type. On the other hand, about 30% of the people examined by Belbin could not be classified at all.

Problems can occur when there is an imbalance between the role types of people in a group. For example, if there are two or more shapers within a group and nobody who takes a chair role to moderate conflicting views, there is likely to be a stormy atmosphere. On the other hand, if a group mainly consists of plants and specialists with no shapers or completer–finishers, the team is likely to have interesting discussions but may not get around to actually implementing anything. When putting together a team Belbin recommends selecting the essential technical specialists first. The group roles of these individuals can then be assessed and any remaining team members can then be allocated with an eye on making the group roles more balanced.

Group performance

The IBM manager was quoted by Angelo Failla in 'Technologies for coordination in a software factory' in *Groupware & Teamwork*, edited by C. U. Ciborra, Wiley & Sons, 1996.

Are groups more effective than individuals working alone? Given the preference of many people attracted to software development for working on their own, this is an important question. In many projects, judgements are needed about which tasks are best carried out collectively and which are best delegated to individuals. As one manager at IBM said: '*Some work yields better results if carried out as a team while some things are slowed down if the work is not compartmentalized on an individual basis.*' Part of the answer lies in the type of task being undertaken.

One way of categorizing group tasks is into:

- Additive tasks
- Compensatory tasks
- Disjunctive tasks
- Conjunctive tasks

Additive tasks mean that the efforts of each participant are added to get the final result, e.g. a gang clearing snow. The people involved are interchangeable.

With *compensatory tasks* the judgements of individual group members are pooled so that the errors of some are compensated for by the inputs from others. For example, individual members of a group are asked to provide estimates of the effort needed to produce a piece of software and the results are then averaged. In these circumstances group work is generally more effective than the efforts of individuals.

Code reviews could be seen as an example of a compensatory task.

With *disjunctive tasks* there is only one correct answer. The effectiveness of the group depends on:

- Someone coming up with the right answer
- The others recognizing it as being correct

Here the group can only be as good as its best member – and could be worse!

Conjunctive tasks are where progress is governed by the rate of the slowest performer. Software production where different staff are responsible for different modules is a good example of this. The overall task is not completed until all participants have completed their part of the work. In this case cooperative attitudes are productive: the team members who are ahead can help the meeting of group objectives by assisting those who are behind. As we will see in a moment, this is an example of *group heedfulness*.

With all types of collective task, but particularly with additive ones, there is a danger of *social loafing*, where some individuals do not make their proper contribution. This can certainly occur with student group activities, but is not unknown in 'real' work environments. As one software developer has commented: *'[The contribution made to others] is not always recognized. Nor is the lack of any contributions... nobody points out those who fail to make any contributions. Like when there's somebody with vital skills and you ask him for help, but he doesn't provide it.'*

The source of the quotation is the paper by Failla that is cited above.

Exercise 12.1

Social loafing is a problem that students often encounter when carrying out group assignments. What steps can participants in a group take to encourage team members to 'pull their weight' properly?

12.3 Decision Making

Before we can look more closely at the effectiveness with which groups can make decisions, we need to look in general terms at the decision-making process.

Many of the evaluation techniques in Chapter 2 are attempts to make decision making more structured.

Decisions can be categorized as being:

- *Structured:* generally relatively simple, routine decisions where rules can be applied in a fairly straightforward way
- *Unstructured:* more complex and often requiring a degree of creativity

Another way of categorizing decisions is by the amount of *risk* and *uncertainty* that is involved.

Some mental obstacles to good decision making

Many of the techniques in Chapter 2 on project selection are based on the rational-economic model.

So far we have rightly stressed a structured, rational, approach to decision makin Many management decisions in the real world, however, are made under pressure a based on incomplete information. We have to accept the role of intuition in such cas but be aware of some mental obstacles to effective intuitive thinking, for example:

- *Faulty heuristics* Heuristics or 'rules of thumb' can be useful but there are dangers:
- they are based only on information that is to hand, which might be misleading;
- they are based on stereotypes, such as accepting a Welshman into a male voice choir without an auditio because of the 'well-known' fact that the Welsh are a great singing nation.
- *Escalation of commitment* This refers to the way that once you have made a decision it is increasingl difficult to alter it even in the face of evidence that it is wrong.
- *Information overload* It is possible to have too much information so that you 'cannot see the wood fo the trees'.

Group decision making

A different type of participatory decision making might occur when end-users are consulted about the way a projected computer system is to operate.

There might be occasions where Amanda at IOE, for instance, might want to consul with her whole project team. With a project team different specialists and points o view can be brought together. Decisions made by the team as a whole are more likel to be accepted than those that are imposed.

Assuming that the meetings are genuinely collectively responsible and have been properly briefed, research would seem to show that groups are better at solving complex problems when the members of the group have complementary skills and expertise. The meeting allows them to communicate freely and to get ideas accepted.

Joint Application Development (JAD) was discussed in Chapter 4.

Groups deal less effectively with poorly structured problems needing creative solutions. Brainstorming techniques can help groups in this situation but research shows that people often come up with more ideas individually than in a group. Where the aim is to get the involvement of end-users of a computer system, then prototyping and participatory approaches such as JAD might be adopted.

Obstacles to good group decision making

Once established, group norms can survive many changes of membership in the group.

Amanda could find that group decision making has disadvantages: it is time-consuming; it can stir up conflicts within the group; and decisions can be unduly influenced by dominant personalities.

Conflict can, in fact, be less than might be expected. Experiments have shown that people will modify their personal judgements to conform to *group norms*, common attitudes developed by a group over time.

You might think that this would moderate the more extreme views that some in the group might hold. In fact, people in groups sometimes make decisions that carry more risk than where they make the decision on their own. This is known as the *risky shift*.

Measures to reduce the disadvantages of group decision making

One method of making group decision making more efficient and effective is by training members to follow a set procedure. The *Delphi technique* endeavours to collate the judgements of a number of experts without actually bringing them face to face. Given a problem, the following procedure is carried out:

- Cooperation of a number of experts is enlisted
- Problem is presented to the experts
- Experts record their recommendations
- These recommendations are collated and reproduced
- Collected responses are recirculated
- Experts comment on the ideas of others and modify their recommendations if so moved
- If the leader detects a consensus then the process is stopped, otherwise the comments are recirculated to the experts

An advantage of this approach is that the experts could be geographically dispersed. However, this means that the process can be time-consuming.

Exercise 12.2

What developments in information technology would be of particular assistance to use of the Delphi technique?

Team heedfulness

Sometimes, despite all these problems, teams work well together. To use the inevitable sporting analogy, a football team does not play at its best when individual players simply display their skills as individuals but do not support one another. A successful move can be triggered where one player sees that another is in a position to score a goal if provided with a ball. This is an example of team heedfulness, where group members are aware of the activities of others that contribute to overall group success and can identify ways of supporting that contribution. In these cases there almost seems to be a 'collective mind'. Clearly there is no such physical entity in reality, and the appearance of a 'collective mind' comes from shared understanding, familiarity and good communications. Some attempts have been made actively to promote this in a software development environment, such as the concept of egoless programming, chief programmer teams and Scrum.

These ideas are explored further in K. Crowston and E. E. Kammerer (1998) 'Coordination and collective mind in software requirements development' *IBM Systems Journal*, 37(2) 227–45.

Egoless programming

In the early days of computer development managers tended to think of the software developer as communing mysteriously with the machine. The tendency was for programmers to see programs as being an extension of themselves and to feel overprotective towards them. The effects of this on the maintainability of programs can be imagined. Gerald Weinberg made the then revolutionary suggestion that programmers and programming team leaders should read each others' programs. Programs would

G. M. Weinberg (1998) *The Psychology of Computer Programming*, Silver Anniversary Edition, Dorset House.

become in effect the common property of the programming group and programming would become 'egoless'. Peer code reviews are based on this idea where items produced by individual team members are checked by selected colleagues – see Chapter 13.

Chief programmer teams

Brooks' *Mythical Man-Month* has already been referred to. He was in charge of the huge team that created the operating system for the IBM 360 range.

The larger the development group the slower it becomes because of the increased communication. Thus large time-critical projects tend to have a more formalized, centralized structure. Brooks stressed the need for design consistency when producing a large complex system and how this is difficult where large numbers of people are involved in development. He suggested reducing this number but making the remaining programmers as productive as possible by giving them more support.

The result was the *chief programmer team*. The chief programmer defines the specification, and designs, codes, tests and documents the software. He or she is assisted by a *co-pilot*, with whom the chief programmer can discuss problems and who writes some code. They are supported by an *editor* to write up the documentation drafted by the chief programmer, a *program clerk* to maintain the actual code, and a *tester*. The general idea is that this team is under the control of a single unifying intellect.

The chief programmer concept was used on the influential *New York Times* data bank project where many aspects of structured programming were tried out. In this case each chief programmer managed a senior-level programmer and a program librarian. Additional members could be added to the team on a temporary basis to deal with particular problems or tasks.

The problem with this kind of organization is getting hold of really outstanding programmers to carry out the chief programmer role. There is also the danger of information overload on the chief programmer. There is in addition the potential for staff dissatisfaction among those who are there simply to minister to the needs of the superstar chief programmers.

Extreme programming (XP)

Extreme programming was discussed in Chapter 4.

The new *extreme programming* (XP) concepts have inherited some of these ideas. Most XP practices can be seen as ways of promoting a 'collective mind'. In conventional software development projects, a typical approach to improving communication and coordination is to introduce more documentation. The advocates of XP argue that this is self-defeating. They suggest other, less formal, methods of communication and coordination. Rather than creating separate documents, the key software products, software code and test data, are enhanced. For example, coding is constantly *refactored* (that is, rewritten) and coding standards are followed to make the code clearly convey how the system works. Test cases and expected results are produced before the code, and act effectively as a form of specification. A user representative should be on hand to clarify user needs. The fit between software components is ensured by continual integration testing. Software development by pairs of developers is advocated – this seems to be a new version of the chief programmer/co-pilot relationship.

We will see that while internal group coordination is enhanced, the problem of coordination between teams remains.

Scrum

It would be self-defeating if the practices advocated by agile approaches should themselves become codified, structured and rigid in application. Promoters of agile methods, such as Kent Beck, are the first to stress that different types of project will need different approaches.

Linda Rising and Norman S. Janoff (2000) 'The Scrum software development process for small teams' *IEEE Software* July/August 26–32 provides a good overview of Scrum in practice.

The *Scrum* software development process illustrates some of these points as it has many elements found in agile methods but also has an element of the chief programmer philosophy. The name 'Scrum' comes from rugby scrums and the image of everyone pushing together in a common undertaking. The process was originally designed for new software product development for a competitive market rather than as a commission for a single client. Here, getting something to market before your competitors may be more important than having a comprehensive range of non-essential features. There is no precise specification of the requirements of a particular client, while having a product that is attractive to a number of customers is important. Proposals for features are likely to evolve as ideas are tried out during development.

The Scrum process starts with a systems architecture and planning phase. This has something of the chief programmer approach as a chief architect defines the overall architecture of the product. The required release date for the product and a set of the desired features of the product, each with a priority, would be defined at this stage.

This phase is now followed by a number of *sprints*, each of which typically lasts between one and four weeks. The features that it is hoped can be developed during a sprint are selected. The tasks needed to implement the features are lised. Sprints are carried out by groups, ideally with about seven developers and at a maximum ten. It is possible for Scrum teams to work in parallel on different sprints, but all teams must finish their sprint on the same day.

The progress of a sprint is marked by short (typically 15 minute) meetings each day. During the meeting, team members report on progress with their current task, describing any obstacles they are experiencing. The meeting allows any colleagues who can assist with a problem to come forward. This might be because the co-worker had a similar problem in the past for which they found a solution. Any resulting problem-solving discussions take place after the meeting. These Scrum meetings should promote shared understanding in the group but also help motivate the team as each person's progress is visible to the whole group.

Sprints are time-boxed and at the end of the sprint period some uncompleted, lower-priority features may be held over. Unlike XP, external requirements are frozen during the sprint – it will be recalled that with XP, changes can be requested at any point. However, at the end of the sprint, all sprint teams meet with the other project stakeholders to review the products created. It is at this point that new features could be added, and previous ones deleted or modified. The priority of desired features could be modified. The features to be built in the next sprint are then chosen, and the tasks needed to deliver those features are planned. The sprint process described above is then repeated.

When all the sprints have been completed, there is a final closure phase where tasks like regression and integration testing and the writing of user and training guides take place to create a final package for delivery to market.

Linda Rising and Norman S. Janoff (2000) have described the implementation of Scrum at AG Communications in the USA. Of interest in their account is the evidence of flexibility in implementing the process. One team, for example, decided to have Scrum meetings three times a week, rather than each day. In another case, a

team decided to break the rule that externally imposed changes should be ignored during a sprint, when an unusually severe externally imposed change was clearly unavoidable.

12.4 Organization and Team Structures

Large software development companies are usually organized into departments. Departmentalization of a company may be based on several criteria such as staff specialization, product lines, categories of customers, or geographical location. For example, a certain company at a high-level may be divided into departments such as banking, embedded application, and telecom software development. Such departments are also called verticals. Small companies do not have high-level departmentalization. Therefore, we can view a small company as having only a single department.

Every department usually handles several projects at any time, and each project is assigned to a separate team of developers. The effectiveness of the developers in achieving the project objectives is significantly affected by how a department is organized into teams and how the individual teams are structured. In this context, two important issues that are critical to the effective functioning of every organization are

- *Department Structure:* How is a department organized into teams?
- *Team Structure:* How are project teams structured?

We discuss these issues in the following section.

Department structure

There are essentially three broad ways in which a software development department can be structured, viz., functional, project, and matrix formats.

Functional format

As will be seen later, a functional division could facilitate outsourcing.

In the functional format, the developers are divided into functional groups based on their specialization or experience. In other words, each functional group comprises developers having expertise in some specific task or functional area. For example, the different functional groups of an organization might specialize in areas such as database, networking, requirements analysis, design, testing, and so on. The functional organizational structure is shown in Figure 12.2(b).

Programme management can facilitate better sharing of staff between projects.

Every developer in an organization would belong to some functional group depending on his/her specialization. For carrying out specific activities, different projects borrow developers from the corresponding functional groups (shown using arrows in Figure 12.2). Upon the completion of their activities, the developers are returned to their respective functional groups. The partially completed product passes from one team to another and evolves due to the work done on it by several teams. A functional team working on a project does not physically meet the members of other functional teams who have carried out other parts of the project. Consequently, a team understands the work carried out by the other functional teams solely by studying the documents produced by them. Unless good quality documents are produced by each team, teams working subsequently on the project will find it hard to understand the work already completed. We can therefore say that a functional organization mandates production of good quality documentation.

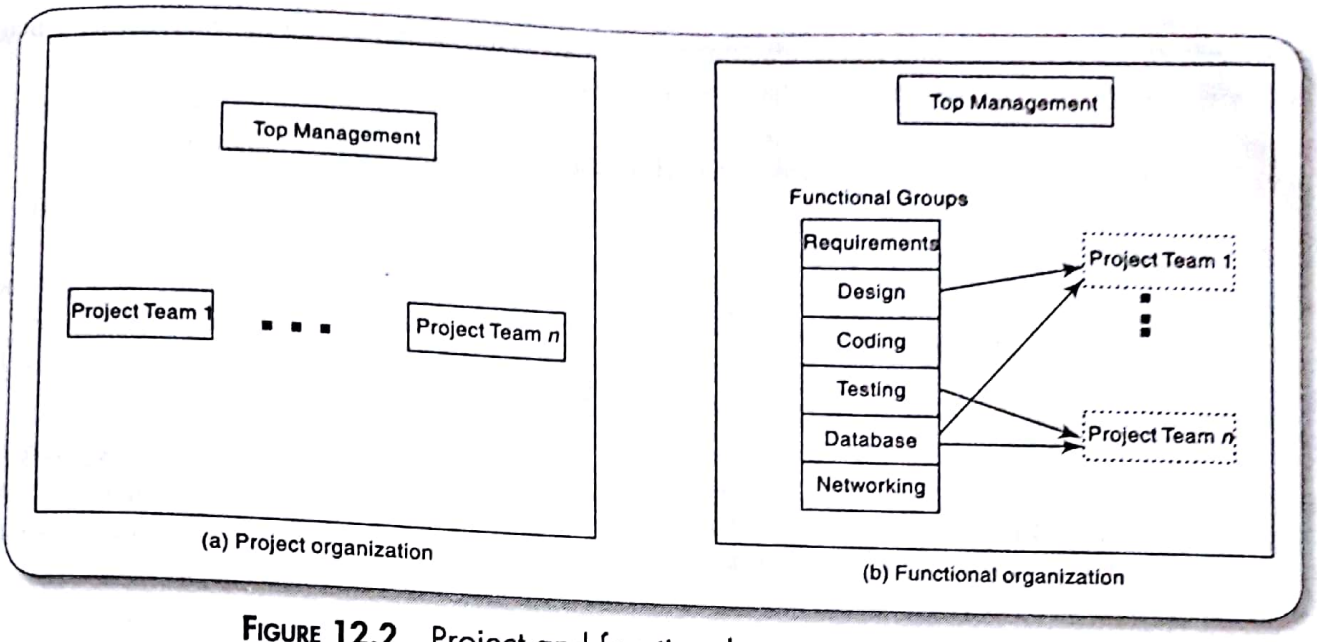

FIGURE 12.2 Project and functional organization structures

Project format

The project format is designed for realizing task-oriented teams. In the project format, at the start of every project, a set of developers are assigned to it (See Figure 12.2(a)). It is assumed that, among them the assigned members can carry out various activities required for project completion. The developers remain with the project till the completion of the project. Thus, the same team carries out all the project activities. This is in contrast to a functional organization, where each developer belongs to a functional group and for completing a project activity, members of the corresponding functional area are assigned to the project temporarily, who are returned back to their respective functional area after completion of the activity.

Functional versus project formats

As we have already pointed out, in a functional organization, the team members carrying out the different activities of a project do not meet each other. The consequent communication gap is a weak point of this format. Also, users often prefer the project team approach because they will have a group dedicated to their needs and they need not deal with members of a large number of functional groups. Further, the project team members build up familiarity with the software over a relatively long period of time and can efficiently carry out maintenance activities. Therefore, the project team approach is advantageous for carrying out the software maintenance activities. On the other hand, the functional format offers several other types of advantages. The main advantages that the functional team format offers are the following.

- *Ease of staffing* The functional organization structure provides an efficient solution to the staffing problem. Usually different numbers of developers are needed to carry out different project activities. The project staffing problem is eased significantly because any number of required personnel can be brought into a project as needed, and they can be returned to the functional group when they complete their work. This possibly is the most important advantage of the functional organization. A project organization structure, on the other hand, mandates the manager to accept a fixed number of developers at the start of the project. These developers work for the entire duration of the project. This results in many team members idling in the initial phases of software development and the entire team comes under tremendous pressure in the later phases of development.

- *Production of good quality documents* A functional organization mandates production of good quality documents, since the team members working on some part of a project do not meet the developers who have completed other parts of the project and gone back to their functional teams.
- *Job specialization* The functional organization structure facilitates developers to become to specialize in particular tasks such as database, networking, compilers, requirements analysis, design, coding, testing, maintenance, etc. They perform these activities again and again for different projects, and thereby gain experience and insights into their respective areas of specialization.
- *Efficient handling of the problems associated with manpower turnover* Functional organizations help to effectively handle the problem of manpower turnover compared to a democratic organization. First, developers are brought in from the functional pool when needed. Also, good documentation produced in this organization structure helps any new member to quickly get familiarized with the work already completed.
- *Career planning* A functional organization makes it easier for a developer to have a career that is technically oriented, called the technical ladder. On the other hand, a project organization tends to facilitate a more general form of career progression, where the developers become business analysts and managers.

Exercise 12.3

In spite of several important advantages of the functional organization, it is rarely adopted by the industry. Explain the apparent paradox.

Matrix format

Matrix format is an extension of a functional format, and is intended to provide the advantages of both the functional and project structures. In a matrix organization, the pool of functional specialists is assigned to different projects as needed. Thus, the deployment of the different functional specialists in different projects can be represented in a matrix (see Figure 12.3). Observe that a member assigned to a project reports to both the manager of his functional group as well as the manager of the project to which he has been assigned. Thus, in a matrix organization, the project manager needs to share the project responsibility with a number of individual functional managers.

Matrix organizations can be characterized as weak or strong, depending upon the relative authority of the functional managers and the project managers. In a strong functional matrix, the functional managers have authority to assign workers to projects and project managers have to accept the assigned personnel. In a weak matrix, the project manager completely controls the project budget, can reject workers from functional groups, and can even decide to hire outside workers.

Though a matrix team organization offers several advantages, two important problems that such an organization may suffer from are the following.

- Due to the multiplicity of authority, conflicts can occur between functional and project managers over allocation of workers.
- In a strong matrix organization, frequent shifting of workers may take place as the functional managers adopt a firefighting mode to tackle the crises in different projects.

Functional group	Project #1	Project #2	Project #3	
#1	2	0	3	Functional manager 1
#2	0	5	3	Functional manager 2
#3	0	4	2	Functional manager 3
#4	1	4	0	Functional manager 4
#5	0	4	6	Functional manager 5
	Project manager 1	Project manager 2	Project manager 3	

FIGURE 12.3 Matrix organization

Team structure

Team structure denotes the reporting, responsibility, and communication structures in individual project teams. We consider only three team structures: democratic, chief programmer and the mixed team organizations, although several other variations to these structures are possible. Please note that it is not necessary that all project teams in an organization are structured the same way. In fact, it is usually the case that within the same organization, different projects adopt different team structures due to differences to their complexities and sizes.

Chief Programmer team

A schematic representation of the chief programmer team structure has been shown in Figure 12.4. In this team structure, a senior member provides the technical leadership and is designated as the chief programmer. The structure of the chief programmer team is the philosophy behind the chief programmer team is in keeping with the suggestions of Brooks, who suggested that the design activity should be carried out by a small team to maintain design consistency. He argued in favour of making the designers as productive as possible through support from the other team members. The chief programmer defines the specification and constructs the high-level design; and then partitions the remaining tasks of detailed design, viz., coding, testing, documentation, etc., into many smaller tasks; and assigns them to the team members. He/she also verifies and integrates the work completed by different team members.

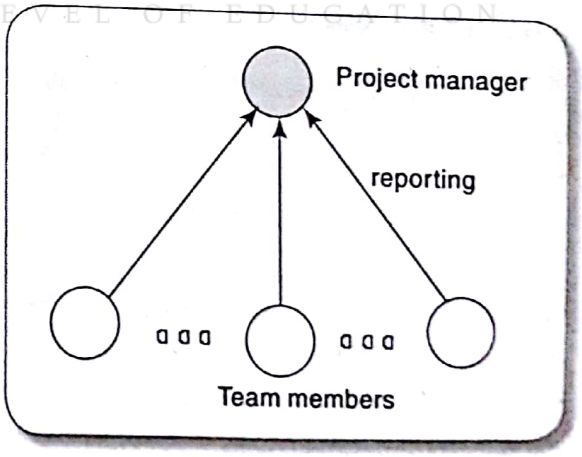

FIGURE 12.4 Chief programmer team structure

Advantages of the chief programmer team

A chief programmer team is more efficient than a democratic team for completing simple and small projects since the chief programmer can quickly work out a satisfactory design and assign work to the team members to code and test different modules of his design solution.

Disadvantages of the chief programmer team

- The chief programmer is provided with an authority to assign work to the team members and to monitor their work. This however leads to lower team morale, as the team members work under the constant supervision of the chief programmer.
- The chief programmer team structure inhibits collective and original thinking by the team members and the chief programmer typically takes all important decisions by himself/herself.
- The chief programmer team is subject to single point failure since too much responsibility and authority is assigned to the chief programmer. That is, a project might get severely affected if the chief programmer either leaves the organization or becomes unavailable for some other reasons.
- A major problem with the chief programmer structure is getting hold of a really outstanding programmer for the role of the chief programmer. Since the chief programmer carries out many tasks individually, there is a danger of information overload on the chief programmer. This is especially true for large projects.

Let us now try to understand the types of projects for which the chief programmer team structure would be appropriate. Suppose an organization has successfully completed many simple MIS projects. Then, for a similar MIS project, chief programmer team structure can be adopted. The chief programmer team structure works well when the task is within the intellectual grasp of a single individual. However, even for small projects the chief programmer team structure should not be used unless the importance of early completion of the project outweighs other factors such as team morale, personal developments, etc.

Democratic team

The democratic team structure, as the name implies, does not enforce any formal team hierarchy. Decisions are taken based on discussions, where any member is free to discuss with any other member as shown in Figure 12.5. Typically, a manager provides the administrative leadership. At different times, different

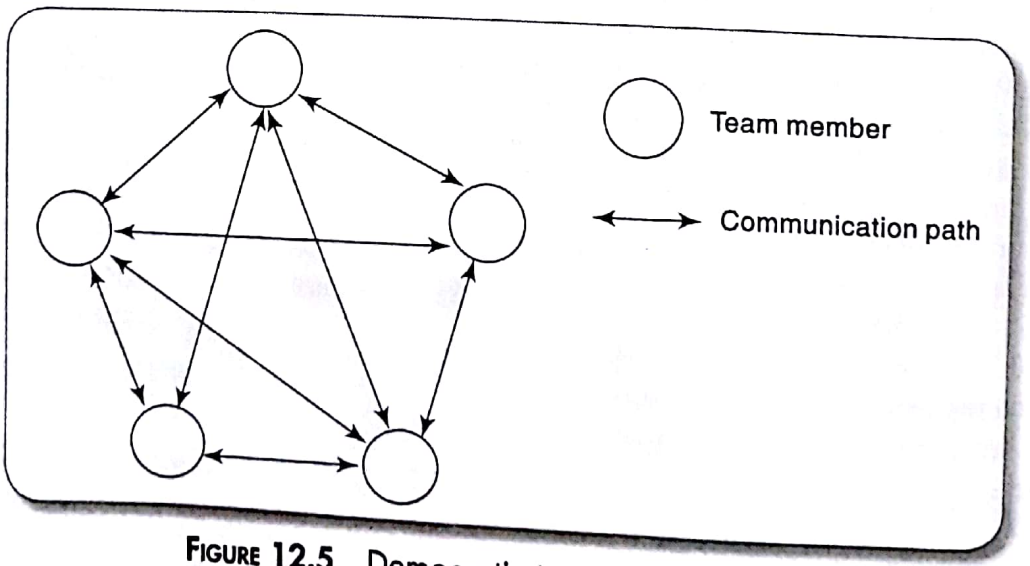

FIGURE 12.5 Democratic team structure

members of the team provide technical leadership. Since a lot of debate and discussions among the team members takes place, for large team sizes significant overhead is incurred on this count.

It is generally accepted that a democratic structure offers higher moral and job satisfaction to the team members. Consequently, a democratic team usually suffers from lower manpower turnover compared to the chief programmer team. Though democratic teams are less productive compared to the chief programmer team for small and simple projects, the democratic team structure is appropriate for less understood problems, since a group of developers can invent better solutions than a single individual as in a chief programmer team. A democratic team structure is suitable for research-oriented projects requiring less than five or six developers. For large sized projects, a pure democratic organization tends to become chaotic. The democratic team organization encourages egoless programming as programmers can share and review one another's work. We have already discussed egoless programming in this section.

Mixed control team structure

The mixed team structure, as the name implies, draws ideas from both the democratic and chief-programmer team structures. The mixed control team structure is shown pictorially in Figure 12.6. In Figure 12.6, the communication paths are shown as dashed lines and the reporting structure is shown using solid arrows. Observe that this team structure incorporates both hierarchical reporting and democratic set up. The democratic arrangement at the level of senior developers is used to decompose the problem into small parts. Democratic setup at the programmer level facilitates working out at effective solution to a single part. This team structure is extremely popular and is being used in many software development companies. The mixed control team organization is suitable for large team sizes.

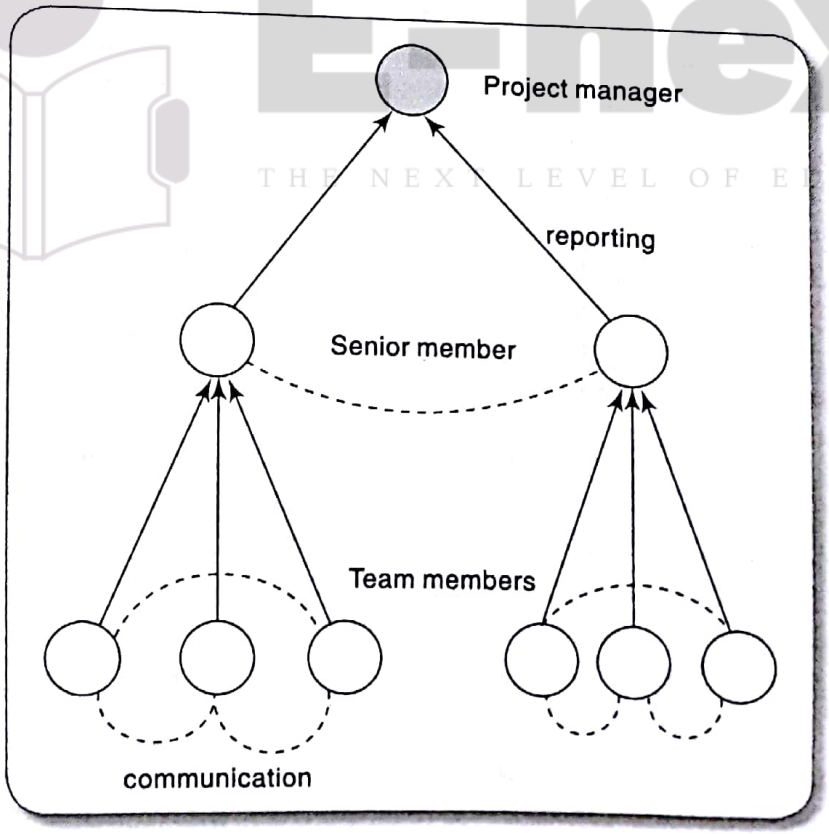

FIGURE 12.6 Mixed control team structure

12.5 Coordination Dependencies

Ian R. McChesney and Séamus Gallagher (2004) 'Communication and co-ordination practices in software engineering projects' *Information and Software Technology* 46 473–89 gives a good introduction to these concepts and insights into their application.

Why and to what extent do the different units within an overall organizational structure really need to communicate? Researchers and innovators in the area of *computer-supported cooperative work* (CSCW) have been interested in identifying the types of coordination where computer tools could be of assistance. A *coordination theory* has been developed which provides a useful classification of coordination dependencies that are likely to exist in any substantial organizational undertaking. These are listed below.

- *Shared resources.* An example in software development projects is where several projects need the services of particular types of scarce technical experts for certain parts of the project. The unavailability of these experts because of commitments elsewhere could delay a project. We noted in Chapter 2 that programme management may be established at a higher level than individual projects to reduce these resources clashes.
- *Producer–customer ('right time') relationships.* A project activity may depend on a product being delivered first. For example, a business analyst may need to produce an agreed requirements document before the development of software components can begin. The Product Flow Diagram (PFD) promoted by the PRINCE2 methodology and described in Chapter 3 can help identify some of these dependencies, but the key point is that some other organizational unit, which could in fact be outside the business, is involved.
- *Task–subtask dependencies.* In order to complete a task a sequence of subtasks have to be carried out. Like the producer–customer relationships described above, this could be reflected in the PFD. Unlike the producer–customer relationship, this sequencing is forced by the technical nature of the thing being created, or the method that is being adopted, rather than by decisions about who is to do what.
- *Accessibility ('right place') dependencies.* This type of dependency is of more relevance to activities that require movement over a large geographical area. The problems of getting an available ambulance to the site of a medical emergency as quickly as possible would a prime example of this. In ICT and software development the examples are less obvious, but arranging the delivery and installation of ICT equipment might be identified as such.
- *Usability ('right thing') dependencies.* In this context, this is a broader concern than the design of user interfaces more usually associated with the term 'usability'. It relates to the general question of *fitness for purpose,* which includes the satisfaction of business requirements. In software development this could lead to activities, such as prototyping, which ensure that the system being created will meet the users' operational needs. It could also involve change management activities when users need to change requirements because of, for example, events in the business environment such as legislative changes.
- *Fit requirements.* This is ensuring that different system components work together effectively. Integration testing is one mechanism that ensures that these requirements are met. One concern of configuration management (as described in Chapter 9) is assessing whether changes made to one component will have knock-on effects on other components.

In many cases, information systems tools can support these coordination tasks. For example, project planning tools, such as Microsoft Project, can be used to help decision making about the allocation of resources both within projects and across a portfolio of concurrent projects. Such tools, through the support they give to the development, analysis and manipulation of activity networks (as we have seen in Chapters 6 and 8), can help

the control of producer–customer and task–subtask dependencies. Another example of software tool support is the use of change management and configuration management databases to keep track of changes to the system under development and thus support usability and fit dependencies.

Ian McChesney and Séamus Gallagher have published a research report which analyses two real-world project environments in terms of the way coordination activities were carried out. Among some quite detailed findings, they noted the use of software tools to support some coordination activities, but also noted how a single person in the project team could have a key coordination role. This person could act as a go-between for staff who need to communicate, for example directing user enquiries to the developer best able to provide information.

E-mail was noted as a principal means of communication. The practice had been developed of copying emails to third parties who might need to be 'kept in the loop', that is, made aware of developments. It is interesting that this development was seen as invaluable – there are reports of other environments where email has become less effective as a means of communication as staff become overwhelmed by the sheer volume of emails.

12.6 Dispersed and Virtual Teams

We have seen how projects require a team of people to carry them out, and the members of this team could each be a specialist in a particular field. Thus, the heart of many projects is *collaborative problem solving*. The Second World War underlined the importance of cooperation between individuals and groups to execute major global operations, and encouraged research after the conflict into effective team working. At that time, group working meant, almost by definition, that the team members worked in close physical proximity. However, in recent years the concept and practice of having *dispersed* or '*virtual teams*' have emerged.

See Tom DeMarco and Timothy Lister (1999) *PeopleWare: Productive Projects and Teams*, 2nd edition, Dorset House.

We have also to see how projects work, especially the development of large software products, needs coordination which in turn means that team members need to communicate. Being located in the same physical space clearly assists this. However, offices can be noisy places and while software development needs communication it also needs periods of solitary concentrated effort. DeMarco and Lister describe the condition of deep concentration needed for effective creative work as 'flow', and suggest that about 15 minutes of uninterrupted effort is needed to achieve this state. Every interruption destroys flow and requires another 15 minutes for its recovery.

It was noted earlier that project managers often have little control over who will be allocated to their team. They are even less likely to have control over the physical environment in which the team will work. Many years ago, research at IBM suggested that ideally each software developer should have:

- 100 square feet of dedicated space
- 30 square feet of work surface
- Noise protection in the form of enclosed offices or partitions at least six feet high

For various reasons it is often difficult for software developers to be provided with this kind of accommodation, yet DeMarco and Lister found clear links between reported noise levels in the workplace and the number of defects found in the resulting software.

One answer is to send the software developers home. One recent (2004) survey found that 77% of businesses allowed at least some of their staff to work at home. Most of those working at home reported that they were more productive. The development of cheap internet-based communications, supported by broadband channels, has reduced the coordination problems that were the drawback of home working.

The survey was carried out by the Economist Intelligence Unit and was reported on in 'Remote working boosts productivity' in *Computing*, 25 November, 2004, p. 6.

Modern communication technologies also mean that organizations can more easily form temporary teams to carry out specific projects from amongst their employees without having to relocate them. The nature of the work carried out in some projects means that the demand for certain specialist skills is intermittent. Ideally the project manager would like to have access to these skills for a short time but then be able to release them and thus avoid further costs. An example of this might be the passing need for a graphic designer to produce aesthetically pleasing designs for a web application project. This desire for flexible labour means that contract workers are often used. The internet allows these contractors to carry out well-defined tasks at their own premises without necessarily having to travel to their clients' site.

It is then only a short step to use 'off-shore' staff who live and work in a different part of the world. Hence we arrive at the dispersed or virtual team.

To recap the possible advantages of such an arrangement:

- A reduction in staff costs by using labour from developing countries where salaries are lower.
- A reduction in the overheads that arise from having your own employees on site, including costs of accommodation, social security payments, training.
- The flexible use of staff – they are not employed when they are not needed.
- Productivity might be higher.
- Use of specialist staff for specific jobs, rather than more general project workers, might improve quality.
- Advantage can be taken of people working in different time zones to reduce task durations – for example, software developers can deliver new versions of code to testers in a different time zone who can test it and deliver the results back at the start of the next working day.

Some of the challenges of dispersed working are:

One estimate was that 24,000 IT and software development jobs would be moved off-shore from the UK between 2003 and 2005, according to a British Computer Society working party report, *Offshoring – a challenge or opportunity for IT Professionals*, (British Computer Society 2004).

- The requirements for work that is distributed to contractors have to be carefully specified.
- The procedures to be followed will need to be formally expressed, where previously practices might have been picked up through observation and imitation of co-workers on site.
- Coordination of dispersed workers can be difficult.
- Payment methods may need to be modified to make them fixed price or piece-rate.
- There may be a lack of trust of co-workers who are remote and never seen.
- Assessment of the quality of delivered products will need to be thorough.
- Different time zones can cause communication and coordination problems.

12.7 Communication Genres

So far we have focused on some of the reasons why men and women engaged on a common project would need to communicate, not only within work groups but with colleagues in other parts of the organization, or in some cases in partner organizations. We have also mentioned various methods of communication but have not examined communications media in a structured way.

Some work has been done to study the characteristics of various methods of communication. One approach has been to try to identify *communication genres*. These refer to something more than just the technical means of communicating. They are types of communication that people are in the habit of making and where there are some 'ground rules' about when and how such communications should be carried out. For some types of communication, such as official meetings, these rules might be quite formal. Within the general heading of 'management meetings' there could be a range of types of meeting each with their own conventions. These could be seen as genres in their own right.

Within the general heading of the email communication genre, advanced email-based applications can be developed where the content of the email is structured by preset proformas designed to fulfil a standard process, such as requesting a change to a software specification. These applications can be seen as communication genres in their own right.

A major influence on the nature of communication genres is the constraints of time and place. Modes of communication can be categorized as combinations of two opposites: same time/different time and same place/different place – see Table 12.1.

TABLE 12.1 Time/place constraints on some communication methods

	Same place	Different place
Same time	Meetings	Telephone
	Interviews etc.	Instant messaging
Different times	Noticeboards	E-mail
	Pigeon-holes	Voicemail
		Documents

The nature of the information to be conveyed also needs to be considered.

- What is the extent and complexity of the information to be conveyed? A telephone conversation may be a very quick way of conveying simple messages – but has disadvantages as a medium for large amounts of information.
- Is it easy to understand? For example, is the context well known to both the sender and the recipient? If it is likely that the recipient will need clarification of aspects of the information, then a mode of communication that is two-way would be best.
- Where the communication is personally sensitive, then face-to-face contact is the most effective form of communication, even if it can be uncomfortable or inconvenient for those concerned.

Exercise 12.4

What would be the best mode of communication for the following?

(a) A developer needs clarification of what is meant by a particular term used in a specification.

(b) The users of a system have found what appears to be a fault in a software application that they use.

(c) A finance director needs to ensure that a software application is changed to conform with new legal requirements.

(d) A software developer can only complete her software component if another developer finishes his component first, but he has not. The first developer is under a lot of pressure from the client to complete the work.

At different stages of a project it is likely that different communication genres will be preferred.

Early stages of a project

Charles Handy (1995) 'Trust and the virtual organization' *Harvard Business Review*, May/June 45–50.

At the start of the project, team members will need to build up their trust and confidence in their co-workers. For this same time/same place communication, actually meeting is probably best. It is argued that this is especially the case with dispersed projects: as Charles Handy has written: '*paradoxically, the more virtual an organization becomes, the more its people need to meet in person*'.

Julian Baggini (2005) 'Touched by your absence' *The Guardian* 6 January.

However, not everyone might agree with this. Julian Baggini, describing his experience of writing a textbook on philosophy with someone he had never met, argues: '*It is simply not necessary to know the whole person in order to have a good relationship with them.... If you get to know and dislike the way that someone behaves outside the office, that can make you uncomfortable with them in it.*'

The early stages of a project are when at least some of the team will be involved in making decisions about the overall design of the product to be delivered and the method by which it is to be created. In the preliminary stages at least, same time/same place meetings are the most effective means of progressing.

Intermediate design stages of the project

Once the overall architecture of the product has been established, detailed design on different components could well be carried out in parallel in different locations. However, some points will need to be clarified and for this, same time/different place communication could well be best, such as the use of teleconferencing.

Implementation stages of the project

M. L. Mazneski and K. M. Chudoba (2000) 'Bridging space over time global virtual team dynamics and effectiveness' *Organization Science*, 11 (5).

Once the design is clarified and everyone knows his or her role, work can progress. Where there is a need to exchange information at this point, different time/different place communication media such as e-mail are likely to be sufficient.

Even at this stage some recommend regular face-to-face meetings of at least some staff as this supplies a rhythm to the project which helps coordination of the project

and maintains motivation. Martha Maznevski and Katherine Chudoba observed the workings of a dispersed project and have noted '*interaction between coordination meetings was mainly in response to the previous meeting or in anticipation of the next one. The coordination meeting served as a heartbeat, rhythmically pumping new life into the team's processes. . . .*'.

12.8 Communication Plans

Communication is important in all projects but a vital matter in the case of dispersed projects. Because of this, consideration of the way that project stakeholders will communicate ought to be a part of the project planning process. Some have gone as far as to suggest that a specific planning document ought to address communication issues affecting the project, not just for dispersed projects but for any project with a substantial number of important stakeholders.

The first step in producing such a plan is to list the main stakeholders, with special attention to those who are participating in the development and implementation of the project – it may be recalled that the identification of stakeholders and their concerns was stressed in Chapter 1 as being fundamental to project success. Once the overall project plan has been created using a process such as that described in Chapter 3, each of the main activities and milestones is examined to see which channels and methods would be best for effective communication with these stakeholders. We have already carried out such a process in our discussion of the communication needs of the different phases of a dispersed project. Consultation with the representatives of the various groups of stakeholders would be an essential part of this process.

The results of this process could be documented in a table with the following column headings.

This table is based on that in the Princeton Project Methodology.

- *What* This contains the name of a particular communication event, for example 'kick-off meeting', or a communication channel, for example 'project intranet site'.
- *Who/target* The target audience for the communication. 'Target audience' may not convey quite the right idea as this implies that there are passive recipients of information from a central authority. In fact the communication event or channel could be a means of eliciting information from the 'audience'.
- *Purpose* What the communication is to achieve.
- *When/frequency* If the communication by means of a single event, then a specific date can be supplied. If the event is a recurring one, such as a progress meeting, then the frequency should be indicated.
- *Type/method* The nature of the communication, for example a meeting or a distributed document.
- *Responsibility* The person who initiates the communication.

12.9 Leadership

See Robert Johansen *et al.* (1991) *Leading Business Teams*, Addison-Wesley.

When Amanda and Brigette first took on project management responsibilities, one of their private anxieties might well have been that staff would not take them seriously. Leadership is generally taken to mean the ability to influence others in a group to act in a particular way to achieve group goals. A leader is not necessarily a good manager or vice versa, as managers have other roles such as organizing, planning and controlling.

Authorities on this subject have found it difficult to agree a list of the common characteristics of good leaders. It would, however, seem safe to say that they seem to have a greater need for power and achievement and have more self-control and more self-confidence than others.

Leadership is based on the idea of authority or power, although leaders do not necessarily have much formal authority. Power may come either from the person's position (*position* power), from the person's individual qualities (*personal* power) or may be a mixture of the two. Position power has been further analysed into:

- *coercive* power, the ability to force someone to do something by threatening punishment;
- *connection* power, which is based on having access to those who have power;
- *legitimate* power, which is based on a person's title conferring a special status;
- *reward* power, where the holder can give rewards to those who carry out tasks to his or her satisfaction.

These ideas are associated with the work of J. R. P. French and B. H. Raven.

Personal power, on the other hand, can be further analysed into:

- *expert* power, which comes from being the person who is able to do a specialized task;
- *information* power, where the holder has exclusive access to information;
- *referent* power, which is based on the personal attractiveness of the leader.

Exercise 12.5

What kinds of power (as defined above) would the following people have?

(a) An internal auditor looking at the payroll system at Brightmouth College.

(b) A consultant who is called in to advise International Office Equipment about ways of improving software development productivity.

(c) The principal of Brightmouth College who has told staff that they must accept a new contract or face the sack.

(d) Brigette in respect to the users of the college payroll system.

(e) Amanda in respect of the people in the project team developing the annual maintenance contract application.

Leadership styles

Amanda and Brigette might be initially concerned about establishing their personal authority. Balanced against this is the need to involve the staff in decision making in order to make the best use of expertise and to gain commitment. Amanda and Brigette will need to judge when they must be authoritative and insist on things and when they must be more flexible and tolerant. Amanda, for example, may decide to be very democratic when formulating plans, but once the plans have been agreed, to insist on a very disciplined execution of the plan. Brigette, on the other hand, may find at Brightmouth College that she alone has the technical expertise to make some decisions, but, once she has briefed staff, they expect to be left alone to get on with the job as they see fit.

This approach is associated with Rensis Likert.

Attempts have been made to measure leadership styles on two axes: directive vs. permissive and autocratic vs. democratic:

- *directive autocrat:* makes decisions alone, close supervision of implementation;
- *permissive autocrat:* makes decisions alone, subordinates have latitude in implementation;

- *directive democrat:* makes decisions participatively, close supervision of implementation;
- *permissive democrat:* makes decisions participatively, subordinates have latitude in implementation.

Another axis used to measure management styles has been on the degree to which a manager is *task-oriented*, that is, the extent to which the execution of the task at hand is paramount, and the degree to which the manager is concerned about the people around them (*people orientation*). It is perhaps not surprising that subordinates appear to perform best with managers who score highly in *both* respects.

It should be emphasized that there is no one best style of management – it depends on the situation.

Work environments vary with amount of control exerted over work. Some jobs are routine and predictable (e.g. dealing with batched computer output). Others are driven by outside factors (e.g. a help desk) or are situations where future direction is uncertain (e.g. at the early stages of a feasibility study). With a high degree of uncertainty subordinates will seek guidance from above and welcome a task-oriented management style. As uncertainty is reduced, the task-oriented manager is likely to relax, becoming more people-oriented, and this will have good results. People-oriented managers are better where staff can control the work they do, without referring matters to their line managers. It is then argued that if control becomes even easier the people-oriented manager will be tempted to get involved in more task-centred questions, with undesirable results.

Research also shows that where team members are relatively inexperienced, a task-oriented approach is most effective. As group members mature, consideration for their personal needs and aspirations becomes more valued. Where maturity is very high, there is no need for a strong emphasis on either of these approaches.

Exercise 12.6

What in your view would be the most appropriate management style when dealing with the following subordinates?

(a) At Brightmouth College, a former member of the local authority who has dealt with the college payroll for several years and who has been employed by the college to set up and manage the new payroll section.

(b) At IOE, a new trainee analyst/programmer who has just joined Amanda's group.

(c) At IOE, a very experienced analyst/programmer in their 40s, who was recruited into the software development department some time ago from the accounts department and who has been dealing with system support for the old maintenance accounts system that is now being revised.

Conclusion

Some of the important points that have been made in this chapter are:

- Consideration should be given, when forming a new project team, to getting the right mix of people and to planning activities which will promote team building
- Group working is more effective with some types of activity than others
- The people who need to communicate most with each other should be grouped together organizationally

- Different styles of leadership are needed in different situations
- Care should be taken to identify the most effective way of communicating with project participants at key points in the project

Further Exercises

1. To what extent is the Belbin approach to balanced teams compatible with having chief programmer teams?
2. If you have been involved recently in a group activity or project, try to categorize each participant according to the Belbin classification. Were there any duplications or gaps in any of the roles? Did this seem to have any impact on progress?
3. Three different mental obstacles to good decision making were identified in the text: faulty heuristics, escalation of commitment and information overload. What steps do you think can be taken to reduce the danger of each of these?
4. Exercise 12.5 asked you to identity the management style most appropriate for each of three different situations. Go back and consider how you as a manager would respond to each of these three situations in terms of practical things to do or avoid.
5. Do you agree with the following statement? 'Few, if any, organization in the real world is purely functional, project, or matrix in nature'. Justify your answer.
6. Explain the advantages of a functional organization over a project organization. Also explain why software development houses prefer to use project organization over functional organization.
7. As a project manager, identify the characteristics that you would look for in a software developer while trying to select personnel for your team.
8. For each of the following questions, exactly one of the options is correct. Select the appropriate option.

   (i) Which one of the following types of team organization would be most suitable for a small yet complex project?

   (a) Chief programmer
   (b) Democratic
   (c) Hybrid
   (d) Squat

   (ii) In which one of the following types of organization is team building likely to be most difficult?

   (a) Functional
   (b) Projectized
   (c) Matrix
   (d) Hybrid

   (iii) Joy is a software engineer who works for a software development company. The company accepts outsourcing projects from overseas clients. Each time it accepts a project, it sets up a team to handle the project. When the project completes, the team is dissolved and the engineers are assigned to other projects. Which one of the following is the organization structure of Joy's company?

   (a) Matrix
   (b) Projectized
   (c) Functional
   (d) Hybrid