# Unit 9 Team Development and Conflict Management

**Structure:**



## 9.1 Introduction

In the previous unit, we have discussed about configuration management. In this unit we shall discuss about an important requirement in project management viz., building a strong development team. We will also learn about various team organizations and how to resolve conflict among team members.

**Objectives:**

After studying this unit, you should be able to:

- explain various team organizations
- assess the performance of different team organizations
- enforce team discipline and resolve conflict among team members

## 9.2 Basic Concepts

In software engineering and other design activities, however, it is not labor, but intellectual power that is in short supply. It is much more difficult, therefore, to deal with deviations from the schedule. The choices of the software engineering manager are rather limited, but there are some options that

must be considered carefully. While it is important to recognize that merely adding engineers is not going to help meet the schedule, it is as important to recognize that the right engineering talent can help. Thus, temporarily reassigning senior engineers to a part of the project that is suffering or hiring expert troubleshooting consultants are viable options. Another option is to examine the requirements and remove the ones that are not absolutely necessary. Sometimes, the requirements are "gold plated"; that is, there is too much attempt to provide a shiny veneer that does not add to the substance of the product. Cutting out unnecessary requirements is called **requirements scrubbing**.

The success of such a recovery action is directly related to the incremental principle. Even if we are not planning incremental delivery of the product, it is important to design a product that may be divided into subsets. Furthermore, it is the key that development proceeds incrementally in the order of importance of the requirements. It will not be helpful if we decide that we can scrub some requirements in order to recover from a slip in the schedule, but those requirements have already been implemented. Thus, the incremental principle must be followed not only during requirements analysis, but also during project planning, scheduling, and implementation.

It is possible that after we consider all options for recovering from a slip in the schedule, we find no appropriate solution. At this point, the best thing to do is to admit the incorrectness of the original plans and schedules and revise the schedule on the basis of new knowledge about the difficulties of the tasks, the capabilities of the engineers, and the availability of the resources. The manager must view a schedule as the best attempt to predict the development cycle. Still, it is merely a prediction, and while it is important to attempt to produce the most accurate prediction possible, it is also important to realize the risks involved with predictions and be prepared to revise the plans if necessary. Clearly, the larger the consequences of a slip in the schedule are, the more important it is to plan and schedule carefully.

For example, if a slip will delay the introduction of a product that has been scheduled for months, and invitations have already been sent out to newspaper reporters and security analysts to attend the formal introduction,

almost any attempt to recover from such a slip can be justified. In less dramatic cases, accepting a delay may be the right course of action.

## 9.3 Organization Types

The organizing function of management deals with devising roles for individuals and assigning responsibility for accomplishing project goals. Organization is basically motivated by the need for cooperation when the goals are not achievable by a single individual in a reasonable amount of time. An organizational structure is necessary at any level of an enterprise, whether it is to coordinate the efforts of a group of vice presidents who report to the president of the corporation or to orchestrate the interactions among programmers who report to a common project manager.

> The aim of an organizational structure is to facilitate cooperation towards a common goal.

Management theorists and practitioners have studied the effects of organizational structures on the productivity and effectiveness of groups. Because the goal of organization is to encourage cooperation, and because cooperation substantially on human characteristics, many general organizational rules apply to any endeavor, whether it deals with software production or automobile production.

The task of organizing can be viewed as building a team: Given a group of people and a common goal, what is the best role for each individual, and how should responsibilities be divided? Analogies with sports teams are illuminating. A basketball team consists of five players on the floor who are playing the game and another perhaps five players who are sitting on the bench as substitutes. Each player knows his or her role. There is one ball, and at any one time, only one player can have it. On well-organized teams, the patterns of cooperation and their effects are clearly visible. In poorly organized teams or teams with novice players, the lack of patterns of cooperation is clearly visible: When one player has the ball, the other four scream for it to be passed to them. The player with the ball, of course, shoots the ball instead of passing it to anyone.

Some of the considerations that affect the choice of an organizational structure are similar to the factors used in cost estimation models that we

have seen earlier. For example, what constitutes an appropriate organization for a project depends on length of the project. Is it a long-term project or a short, one-shot project? If it is a long-term project, it is important to ensure job satisfaction for individuals, lead high morale that reduces turnover. Sometimes, the best composition for a team is mix of junior and senior engineers. This allows the junior engineers to do the less challenging tasks and learn from the senior engineers, who are busy performing the more challenging tasks and overseeing the progress of the junior engineers.

Because of the nature of large software systems, changing requirements, and difficulties of software specification, it has been observed that adding people to a project late in the development cycle leads to further delays in the schedule. Thus, the issue of personnel turnover is a serious one that must be minimized. On a short-term project, personnel turnover is not as important an issue. On a long-term project, it is essential to allow junior personnel to develop their skills and gain more responsibility as senior personnel move on to other responsibilities.

Another issue affecting the appropriate project organization is the nature of the task and how much communication the different team members need to have among themselves. For example, in a well-defined task such as building a payroll system in which modules and their interfaces have been specified clearly, there is not much need for project members to communicate among each other, and excessive communication in the organization will probably lead to a delay in accomplishing their individual tasks. On the other hand, in a project where the task is not clearly understood, communication among team members is beneficial and can lead to a better solution. Strictly hierarchical organizations minimize and discourage communication among team members; more democratic organizations encourage it.

One of the general considerations in team organization is the appropriate size for the team. On the one hand, a small team leads to a more cohesive design, less overhead, more unity, and higher morale. On the other hand, some tasks are too complex to be solved by a small team. Since we cannot control the size or the complexity of the tasks we have to solve, we need to match the size and organization of the team to the problem. Too few people cannot solve an inherently large problem, but assigning an inherently small

problem to a large team also leads to problems, such as too much overhead, overambitious solutions, and solutions that are too costly.

The direct relationship between program complexity and team size is formalized by the **COCOMO** model, where, given the estimate of the size of the software, we can derive the required number of engineers from a formula. How should these engineers be grouped into teams? In general, a small team suffers less overhead and therefore has more productivity per member. We can summarize the considerations of team size as follows: A team should be large enough, but not too large, and small enough, but not too small. Experimental evidence has shown that the optimal size for programming teams is between three and eight, depending on the task. If more than eight people are needed, one can introduce extra levels of management to keep the span of control of each manager manageable.

The size of a team involved in software development is influenced by the characteristics of the software. If a group of modules exhibits high coupling, assigning the modules to different people will require too much interaction among the programmers. Thus, an appropriate design must be accompanied by an appropriate assignment of tasks to individuals and an appropriate team organization that makes that assignment possible. Rather than dogmatically dictating a team organization, one must have a flexible approach and choose the organization on the basis of the design of the system. Of course, the design must be produced by a team in the first place, and a good way to approach the task of building the team is incrementally: Start with a small team that produces a first set of the requirements and design, form a larger team and produce a first implementation, and then use the results produced by the team to decide whether an iteration of the whole cycle is required (as in the spiral model), with a possible need for a team reorganization.

When, why, and how to reorganize a team requires judgment on the part of the manager and cannot be based solely on the design of the software system. As always, the manager must weigh many factors, including the need to complete the project on schedule, to meet budget constraints, to produce a product that meets quality requirements – such as maintainability, which will reduce the project's overall life cycle costs.

### 9.3.1 Centralized-control team organization

Centralized-control team organization is a standard management technique in well understood disciplines. In this mode of organization, several workers report to a supervisor who directly controls their tasks and is responsible for their performance. Centralized control is based on a hierarchical organizational structure in which number of supervisors report to a "second-level" manager and so on up the chain, to the president of the enterprise.

> In general, centralized control works well with tasks which are simple enough that the one person responsible for control of the project can grasp the problem and its solution.

In this kind of organization, one engineer, known as the chief programmer, is responsible for the design and all the technical details of the project. The chief programmer reports to a peer project manager who is responsible for the administrative aspects of the project. Other members of the team are a software librarian and programmers who report to the chief programmer and are added to the team on a temporary basis when needed. Specialists may be used as consultants to the team. The need for programmers and consultants, as well as what tasks they perform is determined by the chief programmer, who initiates and controls all decisions. The software library maintained by the librarian is the central repository for all the documentation, and decisions made by the team. Figure 9.1 is a graphical representation of the patterns of control and communication supported by this kind of organization.

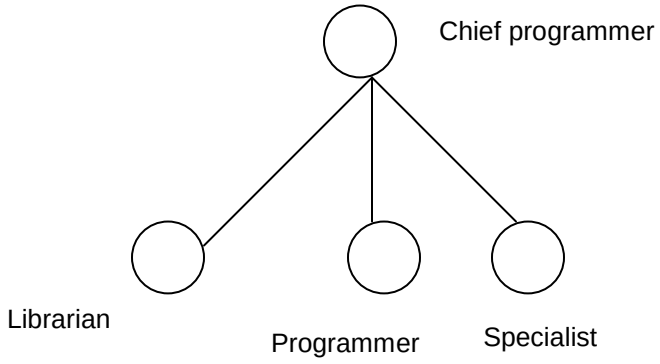


**Fig. 9.1: Chief programmer team**

On the negative side, a chief programmer team has a "single point of failure." Since all communication must go through, and all decisions must be made by, the chief programmer, he or she may become overloaded or, indeed, saturated. The choice of chief programmer is the most important determinant for success of the team.

The success of the chief-programmer team clearly depends on the skill ability of the chief programmer, the size and complexity of the problem.

### 9.3.2 Decentralized-control team organization

In a decentralized-control team organization, decisions are made by consensus, and all work is considered group work. Team members review each other's work and are responsible as a group for what every member produces. Figure 9.2a shows the patterns of control and figure 9.2b shows the communication among team members in a decentralized-control organization. The ring like management structure is intended to show the lack of a hierarchy and that all team members are at the same level.

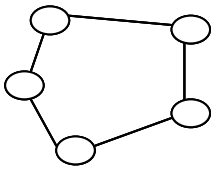

**Fig. 9.2: (a) Management structure**

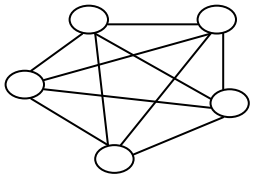

**Fig. 9.2: (b) Patterns of communication**

Such a "democratic" organization leads to higher morale and job satisfaction and, therefore, to less turnover. The engineers feel more ownership of the project and responsibility for the problem, resulting in higher quality in their work.

A decentralized-control organization is more suited for long-term projects, because the amount of intragroup communication that it encourages leads to a longer development time, presumably accompanied by lower life cycle costs.

The proponents of this kind of team organization claim that it is more appropriate for less understood and more complicated problems, since a group can invent better solutions than a single individual can. Such an organization is based on a technique referred to as "egoless programming,"

because it encourages programmers to share and review one another's work.

On the negative side, decentralized-control team organization is not appropriate for large teams, where the communication overhead can overwhelm all the engineers, reducing their individual productivity.

### 9.3.3 Mixed-control team organization

A mixed-control team organization attempts to combine the benefits of centralized and decentralized control, while minimizing or avoiding their disadvantages. Rather than treating all members the same, as in a decentralized organization, or treating single individual as the chief, as in a centralized organization, the mixed organization differentiates the engineers into senior and junior engineers. Each senior engineer leads a group of junior engineers and reports, in its turn, to a project manager. Control is vested in the project manager and senior programmers, while communication is decentralized among each set of individuals, peers, and their immediate supervisors. The patterns of control and communication in mixed-control organizations are shown in figures 9.3a and b.

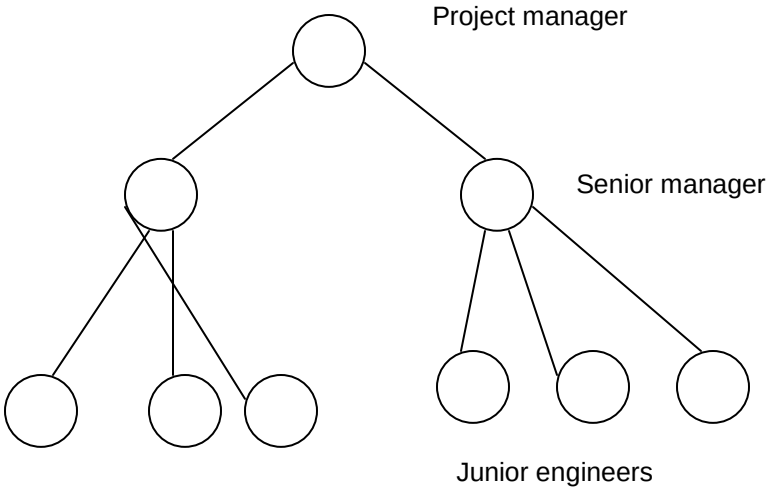


**Fig. 9.3: (a) Management structure**

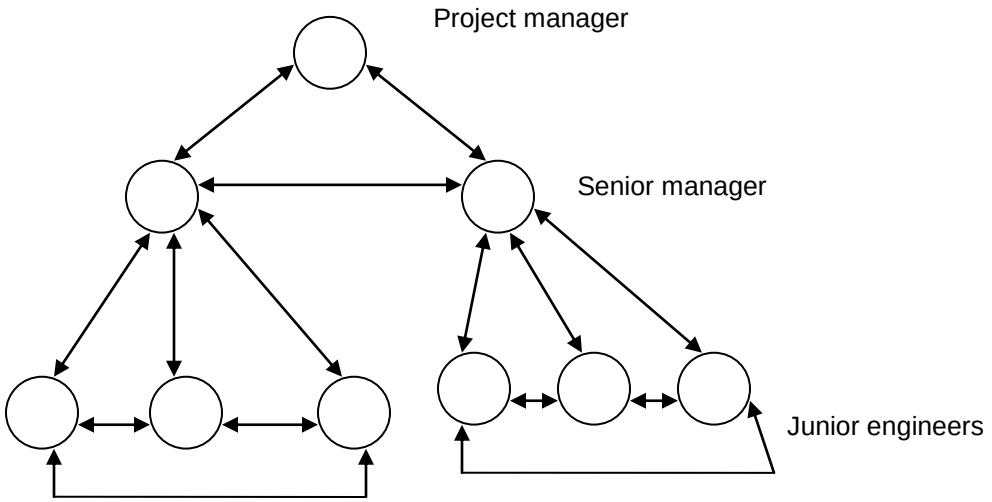


**Fig. 9.3: (b) Patterns of communication**

A mixed-mode organization tries to limit communication to within a group that is most likely to benefit from it. It also tries to realize the benefits of group decision making by vesting authority in a group of senior programmers or architects. The mixed-control organization is an example of the use of a hierarchy to master the complexity of software development as well as organizational structure.

**Self Assessment Questions**

1. The aim of an organizational structure is to facilitate cooperation towards a common goal. (True / False)
2. Cutting out unnecessary requirements in the software project is called _____.
3. _____ team organization attempts to combine the benefits of centralized and decentralized control, while minimizing or avoiding their disadvantages.

## 9.4 Case Study 1: Open-Source Development Team Organization

The open-source development approach is in some sense not suitable to commercial software development projects because of its reliance on (unpaid) volunteer developers and the lack of an organized schedule, but it is useful as a case study because of its ability to develop reliable and useful products.

The team organization is a mixed mode. Each module has a responsible person. Anyone may review the module and send in corrections and other contributions to the responsible person, who is the ultimate arbiter of what goes into the eventual release of the module.

One of the surprising conclusions of the open-source development organization has been that by increasing the number of people on a project, it is indeed possible to achieve more functionality in the product without overwhelming it with communication overhead. The underlying causes of the success of the approach are difficult to assess, but certainly the combination of democratic team organization the appointment of recognized “gurus” to be responsible for key decisions is essential factor.

## 9.5 An Assessment of Team Organizations

In the previous three subsections (9.3.1 to 9.3.3) we presented different ways of organizing software development teams. Each kind of organization discussed has its proponents and detractors. Each also has its appropriate place.

Experimental assessment of different organizational structures is difficult. It is clearly impractical to run large software development projects using two different types of organization just for the purpose of comparing the effectiveness of the two structures. While cost estimation models can be assessed on the basis of how well they predict actual software costs, an organizational structure cannot be assessed so easily, because one cannot compare the results achieved with those one would have achieved with a different organization.

Experiments have been run to measure the effects of such things as the size of the team and the complexity of the task on the effectiveness of development teams. In the choice of team organization, however, it appears that we must be content with the following general considerations:

- Just as no life cycle model is appropriate for all projects, no team organization is appropriate for all tasks.
- Decentralized control is best when communication among engineers is necessary for achieving a good solution.
- Centralized control is best when speed of development is the most important goal and the problem is well understood.

- An appropriate organization limits the amount of communication to what is necessary for achieving the goals of the project—no more and no less.
- An appropriate organization may have to take into account goals other speed of development, including lower life cycle costs, reduced personnel turnover, repeatability of the process, the development of junior engineers into senior engineers, and widespread dissemination of specialized knowledge and expertise among personnel.

## 9.6 Case Study 2: Nokia Software Factories

To cope with the huge growth of its business and the increasing importance of software in the development of mobile phones, the Nokia Corporation organizes its software development teams as "software factories." The Nokia software factories are based on four principles:

- A *geographically distributed environment.* A typical project consists of developers dispersed among three to four sites. Working synchronously is not always possible, because of differences in time zones.
- *Product family architecture.* Architecture is developed for an entire family and components are developed to be used in all family members.
- *Concurrent engineering.* Components are developed concurrently at different sites. To build a given product, the needed components are retrieved from various sites and combined in a central location. The parallel development of components shortens the time to market for the product.
- *The use of tools.* The process is supported by the use of tools, especially requirements engineering, design, coding, version management, configure management, and testing.

## 9.7 Team Discipline

Team discipline consists of the following issues:

- Attendance
- Breaks/late coming/early going
- Punctuality, e.g. in attending meetings
- Late sitting/weekend work/shift work
- Leave planning
- Sharing resources

- Cost consciousness
- Avoiding waste
- Information sharing
- Confidentiality
- Record keeping
- Analytical approach
- Escalation of issues
- Representing organization
- Customer interaction

## 9.8 Conflict Management

A conflict occurs for various reasons. But each conflict should be resolved in proper time. Conflict management refers to the long-term management of intractable conflicts. It is the label for the variety of ways by which people handle grievances standing up for what they consider to be right and against what they consider to be wrong. The following list shows causes, resolution and confrontation.

**Causes**

- Priorities of different interest groups
- Scarcity of resources
- Concurrence on standards and procedures
- Work allocation – needs and expectations
- Personality clashes

**Resolution**

- Withdraw or retreat
- De-emphasize importance of disagreement
- Compromise
- Use force/power

**Confrontation**

- Recognize potential problems
- Try to identify and treat the cause
- Try team approach
- Try analytical approach
- Try emotional approach

**Self Assessment Questions**

4. Decentralized control is best when communication among engineers is necessary for achieving a good solution. (True / False)
5. _____ is best when speed of development is the most important goal and the problem is well understood.
6. _____ refers to the long-term management of intractable conflicts.

## 9.9 Summary

Let's summarize important points discussed in this unit:

- This unit presented a concise view of the management approach towards controlling and managing the human resources, i.e., managing the group of software engineers.
- We have discussed about centralized, decentralized control and mixed control.
- In centralized control, several workers (i.e. software engineers) report to a supervisor who directly controls their tasks and is responsible for their performance.
- In decentralized approach decisions are made by consensus, and all work is considered group work. Team members review each other's work and are responsible as a group for what every member produces.
- A mixed-control team organization attempts to combine the benefits of centralized and decentralized control, while minimizing or avoiding their disadvantages.
- Also we have discussed about team discipline and conflict management. Unless we have team discipline the work cannot be finished in time. Similarly, conflict management also. Conflicts are inevitable. Somewhere around the system conflicts arises. But we have to find out the reasons for the conflict and should sort out properly.
- A manager has to accept the difficulties of the job and carefully evaluate the impact of any newly proposed panaceas.

## 9.10 Terminal Questions

1. What are the basic considerations in developing a software project?
2. Explain the need for various organizational types.
3. What do you mean by centralized team organization? Explain.

4. Give the features of decentralized team organization.
5. List out the advantages of mixed mode control organization.
6. Bring out the features of open source team development.
7. How will you assess various team organizations?
8. What are the features of Nokia Software Factories?
9. List out key characteristics of team discipline.
10. How will you resolve conflict among team members?

## 9.11 Answers

### Self Assessment Questions

1. True
2. Requirements Scrubbing
3. Mixed-Control
4. True
5. Centralized-Control
6. Conflict Management

### Terminal Questions

1. In software engineering design activities, generally intellectual power will be in short supply. It is much more difficult to deal with deviations from the schedule. The choices of the software engineering manager are rather limited, but there are some options that must be considered carefully. While it is important to recognize that merely adding engineers is not going to help meet the schedule, it is as important to recognize that the right engineering talent can help. Thus, temporarily reassigning senior engineers to a part of the project that is suffering or hiring expert troubleshooting consultants are viable options. (Refer Section 9.2 for detail)
2. The organizing function of management deals with devising roles for individuals and assigning responsibility for accomplishing project goals. Organization is basically motivated by the need for cooperation when the goals are not achievable by a single individual in a reasonable amount of time. An organizational structure is necessary at any level of an enterprise, whether it is to coordinate the efforts of a group of vice presidents who report to the president of the corporation or to orchestrate the interactions among programmers who report to a common project manager. (Refer Section 9.3)

3. Centralized-control team organization is a standard management technique in well understood disciplines. In this mode of organization, several workers report to a supervisor who directly controls their tasks and is responsible for their performance. Centralized control is based on a hierarchical organizational structure in which number of supervisors report to a "second-level" manager and so on up the chain, to the president of the enterprise. (Refer Sub-section 9.3.1)
4. In a decentralized-control team organization, decisions are made by consensus, and all work is considered group work. Team members review each other's work and are responsible as a group for what every member produces. (Refer Sub-section 9.3.2)
5. A mixed-control team organization attempts to combine the benefits of centralized and decentralized control, while minimizing or avoiding their disadvantages. Rather than treating all members the same, as in a decentralized organization, or treating single individual as the chief, as in a centralized organization, the mixed organization differentiates the engineers into senior and junior engineers. Each senior engineer leads a group of junior engineers and reports, in its turn, to a project manager. Control is vested in the project manager and senior programmers, while communication is decentralized among each set of individuals, peers, and their immediate supervisors. (Refer Sub-section 9.3.3)
6. In open source development, the team organization is a mixed mode. Each module has a responsible person. Anyone may review the module and send in corrections and other contributions to the responsible person, who is the ultimate arbiter of what goes into the eventual release of the module. (Refer Section 9.4)
7. Experimental assessment of different organizational structures is difficult. It is clearly impractical to run large software development projects using two different types of organization just for the purpose of comparing the effectiveness of the two structures. While cost estimation models can be assessed on the basis of how well they predict actual software costs, an organizational structure cannot be assessed so easily, because one cannot compare the results achieved with those one would have achieved with a different organization. (Refer Section 9.5)

8. The Nokia software factories are based on four principles:
   - A geographically distributed environment.
   - Product family architecture.
   - Concurrent engineering.
   - The use of tools. (Refer Section 9.6)
9. Team discipline consists of the following issues:
   - Attendance
   - Breaks/late coming/early going
   - Punctuality, e.g. in attending meetings
   - Late sitting/weekend work/shift work
   - Leave planning (Refer Section 9.7)
10. A conflict occurs for various reasons. But each conflict should be resolved in proper time. Conflict management refers to the long-term management of intractable conflicts. It is the label for the variety of ways by which people handle grievances standing up for what they consider to be right and against what they consider to be wrong. (Refer Section 9.8)