SOFTWARE PROJECT MANAGEMENT

Effective software project management is crucial to the success of any software project. In the past, several software projects have failed not for want of competent technical professionals neither for lack of resources, but due to the use of faulty software project management practices. Therefore, it is important to carefully learn the latest software project management techniques.

Software project management is a very vast topic. In fact, a course for a full semester can be conducted on effective techniques for software project management. However, in this chapter, we shall restrict ourselves to only some basic issues.

The main goal of software project management is to enable a group of software developers to work efficiently towards successful completion of the project.

Should the responsibility of software project management rest always on a dedicated full-time project manager? Large projects usually have full-time project managers. However, for small software development projects, one of the software developers assumes the responsibilities of software project management in addition to his normal responsibilities.

In this chapter, we discuss the important responsibilities and activities of a software project manager. We start with a discussion on the scope of the work responsibilities of a project manager. Subsequently, we provide an overview of the planning activity and the organization of the project plan document. We then discuss estimation and scheduling techniques. Finally, we provide an overview of the risk and configuration management issues.

3.1 RESPONSIBILITIES OF A SOFTWARE PROJECT MANAGER

In this section, we examine the principal job responsibilities of a project manager and the skills necessary to accomplish these.

3.1.1 Job Responsibilities of a Software Project Manager

Software project managers take the overall responsibility of steering a project to success. This surely is a very hazy job description. But, it is very difficult to objectively describe the job

responsibilities of a project manager. The responsibilities and activities of a project manager is large and varied. The job responsibility of a project manager ranges from invisible activities like building up team morale to highly visible customer presentations. Most managers take responsibility for project proposal writing, project cost estimation, scheduling, project staffing, software process tailoring, project monitoring and control, software configuration management, risk management, interfacing with clients, managerial report writing and presentations, etc. These activities are certainly numerous, varied and difficult to enumerate, but we can broadly classify them into two major types of responsibilities of the project manager.

> We can broadly classify the different activities of a project manager into project planning, and project monitoring and control activities.

We give an overview of these two responsibilities. Later, we discuss them in more detail.

1. Project planning: Project planning involves estimating several characteristics of the project and then planning the project activities based on the estimates made. Project planning is undertaken immediately after the feasibility study phase and before the requirements analysis and specification phase. The initial project plans that are made are revised from time to time as the project progresses and more project data become available.

2. Project monitoring and control activities: The project monitoring and control activities are undertaken once the development activities start. The aim of the project monitoring and control activities is to ensure that the development proceeds as per plan. The plan is changed whenever required to cope up with the situation at hand.

3.1.2 Skills Necessary for Software Project Management

A theoretical knowledge of different project management techniques is certainly necessary to become a successful project manager. However, effective software project management frequently calls for good qualitative judgement and decision-making capabilities. In addition to having a good grasp of the latest software project management techniques such as cost estimation, risk management, and configuration management, etc.; project managers need good communication skills and the ability get work done. However, some skills such as tracking and controlling the progress of the project, customer interaction, managerial presentations, and team building are largely acquired through experience. Nonetheless, the importance of a sound knowledge of the prevalent project management techniques cannot be overemphasized. The objective of the rest of this chapter is to introduce you to the same.

With this brief discussion on the responsibilities and roles of software project managers, in the next section we examine some important issues in project planning.

3.2 PROJECT PLANNING

Once a project is found to be feasible, software project managers undertake project planning. Project planning is undertaken and completed even before any development activity starts. Project planning consists of the following essential activities:

1. **Estimation:** The following project attributes have to be estimated.
   (i) *Cost* How much is it going to cost to develop the software?
   (ii) *Duration* How long is it going to take to develop the product?
   (iii) *Effort* How much effort would be required to develop the product?

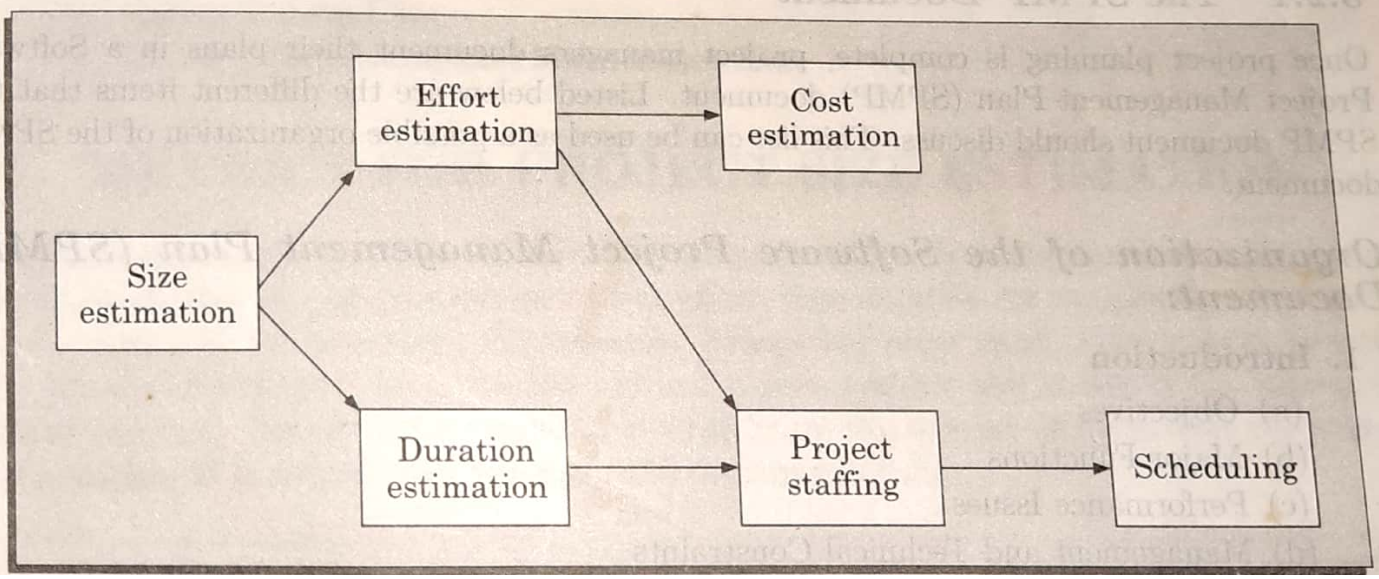
   The effectiveness of all other planning activities such as scheduling and staffing are based on the accuracy of these estimations.

2. **Scheduling:** After the estimations are made, the schedules for manpower and other resources have to be developed.

3. **Staffing:** Staff organization and staffing plans have to be made.

4. **Risk management:** Risk identification, analysis, and abatement planning have to be done.

5. **Miscellaneous plans:** Several other plans such as quality assurance plan, configuration management plan, etc. have to be done.



3.9 SCHEDULING

Scheduling the project tasks is an important project planning activity. It involves deciding which tasks would be taken up when. In order to schedule the project activities, a software project manager needs to do the following:

1. Identify all the tasks needed to complete the project.
2. Break down large tasks into small activities.
3. Determine the dependency among different activities.
4. Establish the most likely estimates for the time durations necessary to complete the activities.
5. Allocate resources to activities.
6. Plan the starting and ending dates for various activities.
7. Determine the **critical path**. A critical path is a chain of activities that determines the duration of the project.

The first step in scheduling a software project involves identifying all the tasks necessary to complete the project. A good knowledge of the intricacies of the project and the development process helps the managers to effectively identify the important tasks of the project. Next, the large tasks are broken down into a logical set of small activities which would be assigned to different developers. The work breakdown structure formalism discussed in section 3.9.1 helps the manager to breakdown the tasks systematically.

After the project manager has broken down the tasks and created the work breakdown structure, he has to find the dependency among the activities. Dependency among the different activities determines the order in which the different activities would be carried out. If an activity A requires the results of another activity B, then activity A must be scheduled after activity B. In general, the task dependencies define a partial ordering among tasks, i.e. each tasks may precede a subset of other tasks, but some tasks might not have any precedence ordering defined between them (called concurrent task). Let us discuss the dependency among the activities are represented in the form of an activity network.

Once the activity network representation has been worked out, resources are allocated to each activity. Resource allocation is typically done using a Gantt chart. After resource allocation is done, a PERT (Project Evaluation and Review Technique) chart representation is developed. The PERT chart representation is suitable for project monitoring and control. The work breakdown structure, activity network, Gantt and PERT charts are discussed further. For task scheduling, the project manager needs to decompose the project tasks into a set of activities. The time frame when each activity is to be performed is to be determined. The end of each activity is called a **milestone.** The project manager tracks the progress of a project by monitoring the the timely completion of the milestones. If he observes that the milestones start getting delayed, then he has to carefully control the activities, so that the overall deadline can still be met.

3.9.1 Work Breakdown Structure

Work Breakdown Structure (WBS) is used to decompose a given task set recursively into small activities. WBS provides a notation for representing the major tasks needed to be carried out in order to solve a problem. The root of the tree is labelled by the problem name. Each node of the tree is broken down into smaller activities that are made the children of the node. Each activity is recursively decomposed into smaller sub-activities until at the leaf level, the activities requires approximately two weeks to develop. Figure 3.7 represents the WBS of an MIS (Management Information System) software.

While breaking down a task into smaller tasks, the manager has to make some hard decisions. If a task is broken down into a large number of very small activities, these can be distributed to a larger number of developers. If the activity ordering permits that solutions to these can be carried out independently. Thus, it becomes possible to develop the product faster (with the help of additional manpower of course!). Therefore, to be able to complete a project in the least amount of time, the manager needs to break large tasks into smaller ones, expecting to find more parallelism. However, it is not useful to subdivide tasks into units which take less than a week or two to execute. Very fine subdivision means that a disproportionate amount of time must be spent on preparing and revising various charts.

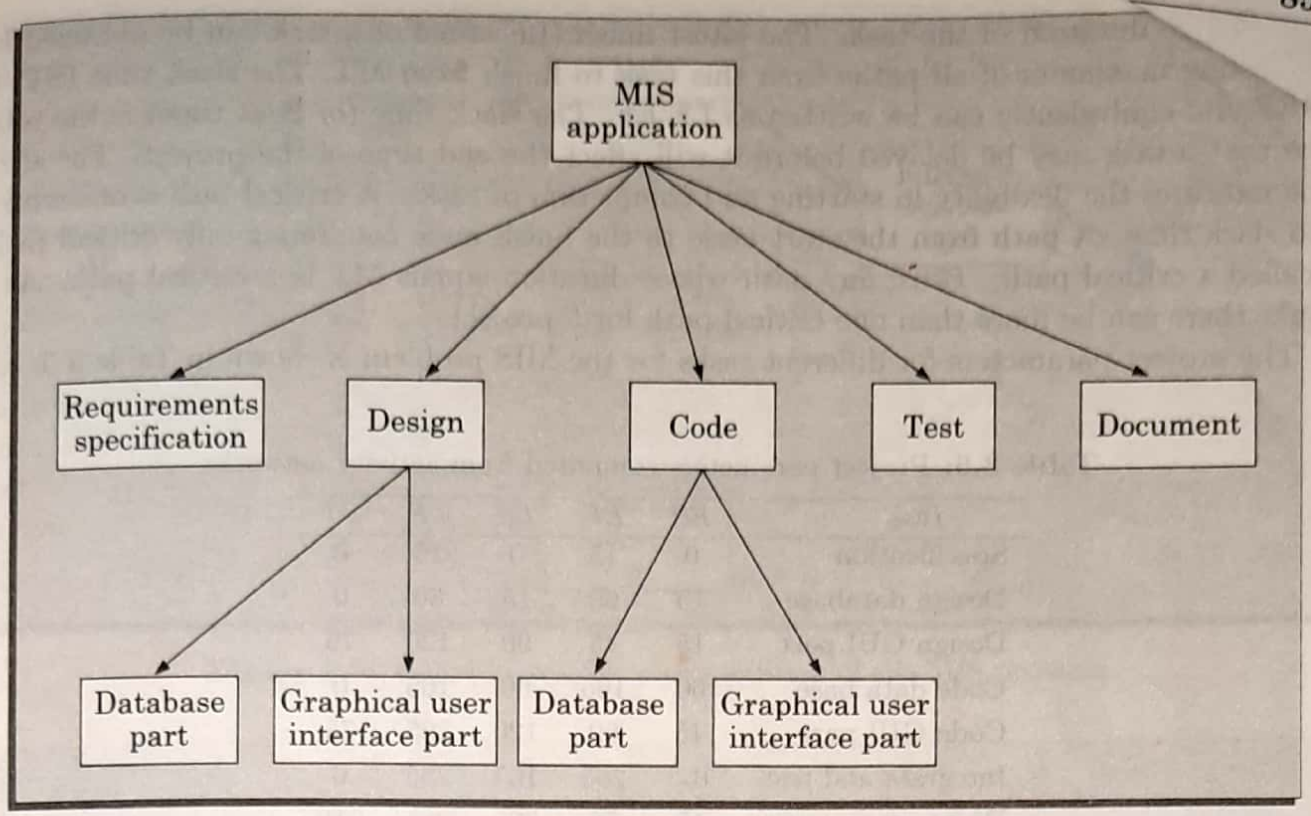


Figure 3.7: Work breakdown structure of an MIS problem.

3.9.2 Activity Networks and Critical Path Method

WBS representation of a project is transformed into an activity network by representing activities identified in WBS along with their interdependencies. An activity network shows the different activities making up a project, their estimated durations, and interdependencies. Each activity is represented by a rectangular node and the duration of the activity is shown alongside each task.

Managers can estimate the time durations for the different tasks in several ways. One possibility is that they can empirically assign durations to different tasks. This however is not a good idea, because software developers often resent such unilateral decisions. However, some managers prefer to estimate the time for various activities themselves. Many managers believe that an aggressive schedule motivates the developers to do a better and faster job. On the other hand, careful experiments have shown that unrealistically aggressive schedules not only cause developers to compromise on intangible quality aspects, but also are a cause for schedule delays. A possible alternative is to let each engineer himself estimate the time for an activity he can be assigned to. This approach can help to accurately estimate the task durations without creating undue schedule pressures.

Critical Path Method (CPM)

From the activity network representation, following analysis can be made. The minimum time (MT) to complete the project is the maximum of all paths from start to finish. The earliest start (ES) time of a task is the maximum of all paths from the start to this task. The latest start time is the difference between MT and the maximum of all paths from this task to the finish. The earliest finish time (EF) of a task is the sum of the earliest start time of the

task and the duration of the task. The latest finish (LF) time of a task can be obtained by subtracting maximum of all paths from this task to finish from MT. The slack time (ST) is LS-EF and equivalently can be written as LF-EF. The slack time (or float time) is the total time that a task may be delayed before it will affect the end time of the project. The slack time indicates the flexibility in starting and completion of tasks. A critical task is one with a zero slack time. A path from the start node to the finish node containing only critical tasks is called a critical path. Thus, any path whose duration equals MT is a critical path. As a result, there can be more than one critical path for a project.

The project parameters for different tasks for the MIS problem is shown in Table 3.6.

Table 3.6: Project parameters computed from activity network

Task	*ES*	*EF*	*LS*	*LF*	*ST*
Specification	0	15	0	15	0
Design database	15	60	15	60	0
Design GUI part	15	45	90	120	75
Code data base	60	165	60	165	0
Code GUI part	45	90	120	165	75
Integrate and test	165	285	165	285	0
Write user manual	15	75	225	285	210

The critical paths are all the paths whose duration equals MT. The critical path in Figure 3.8 is shown with a thick arrow.

3.9.3 Gantt Charts

Gantt charts are mainly used to allocate resources to activities. The resources allocated to activities include staff, hardware, and software. Gantt charts (named after its developer Henry Gantt) are useful for resource planning. A Gantt chart is a special type of bar chart where each bar represents an activity. The bars are drawn along a time line. The length of each bar is proportional to the duration of time planned for the corresponding activity.

Gantt charts used in software project management are actually an enhanced version of the standard Gantt charts. In the Gantt charts used for software project management, each bar consists of a white part and a shaded part. The shaded part of the bar shows the length of time each task is estimated to take. The white part shows the *slack* time, that is, the latest time by which a task must be finished. A Gantt chart representation for the MIS problem of Figure 3.8 is shown in Figure 3.9.

3.9.4 PERT Charts

PERT (Project Evaluation and Review Technique) charts consist of a network of boxes and arrows. The boxes represent activities and the arrows represent task dependencies. PERT chart represents the statistical variations in the project estimates assuming a normal distribution. Thus, in a PERT chart instead of making a single estimate for each task, pessimistic, likely, and optimistic estimates are made. The boxes of PERT charts are usually annotated

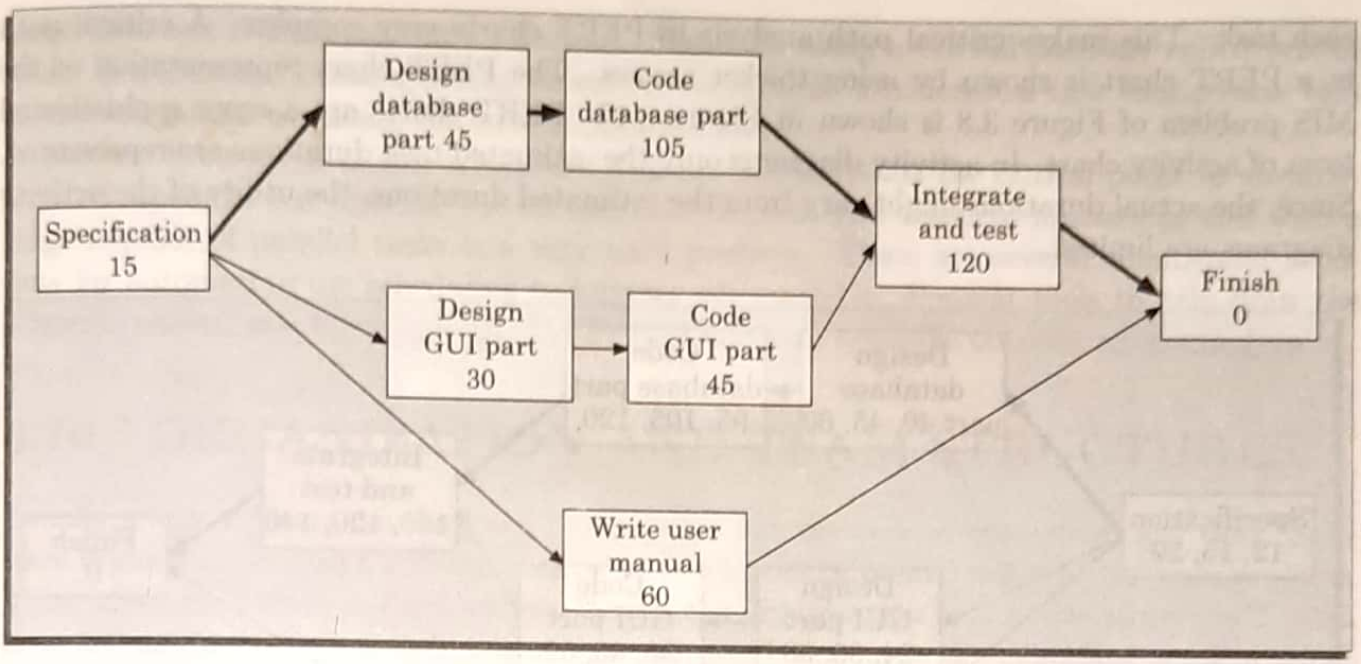


Figure 3.8: Activity network representation of the MIS problem.

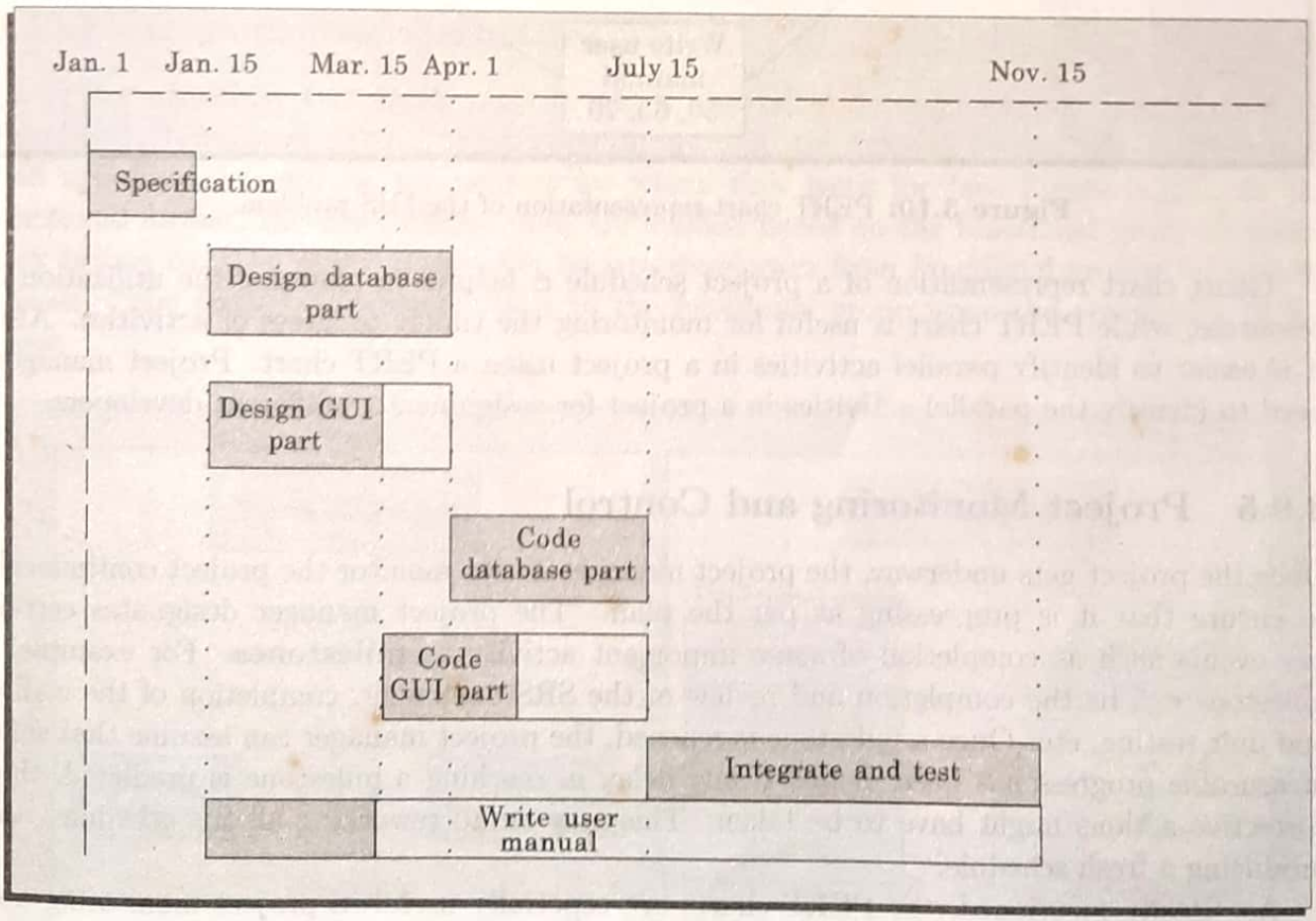


Figure 3.9: Gantt chart representation of the MIS problem.

with the pessimistic, likely, and optimistic estimates for every task. Since all possible completion times between the minimum and maximum duration for every task has to be considered, there is not one but many critical paths, depending on the permutations of the estimates for

ach task. This makes critical path analysis in PERT charts very complex. A critical path a PERT chart is shown by using thicker arrows. The PERT chart representation of the IIS problem of Figure 3.8 is shown in Figure 3.10. PERT charts are a more sophisticated rm of activity chart. In activity diagrams only the estimated task durations are represented. ince, the actual durations might vary from the estimated durations, the utility of the activity iagrams are limited.

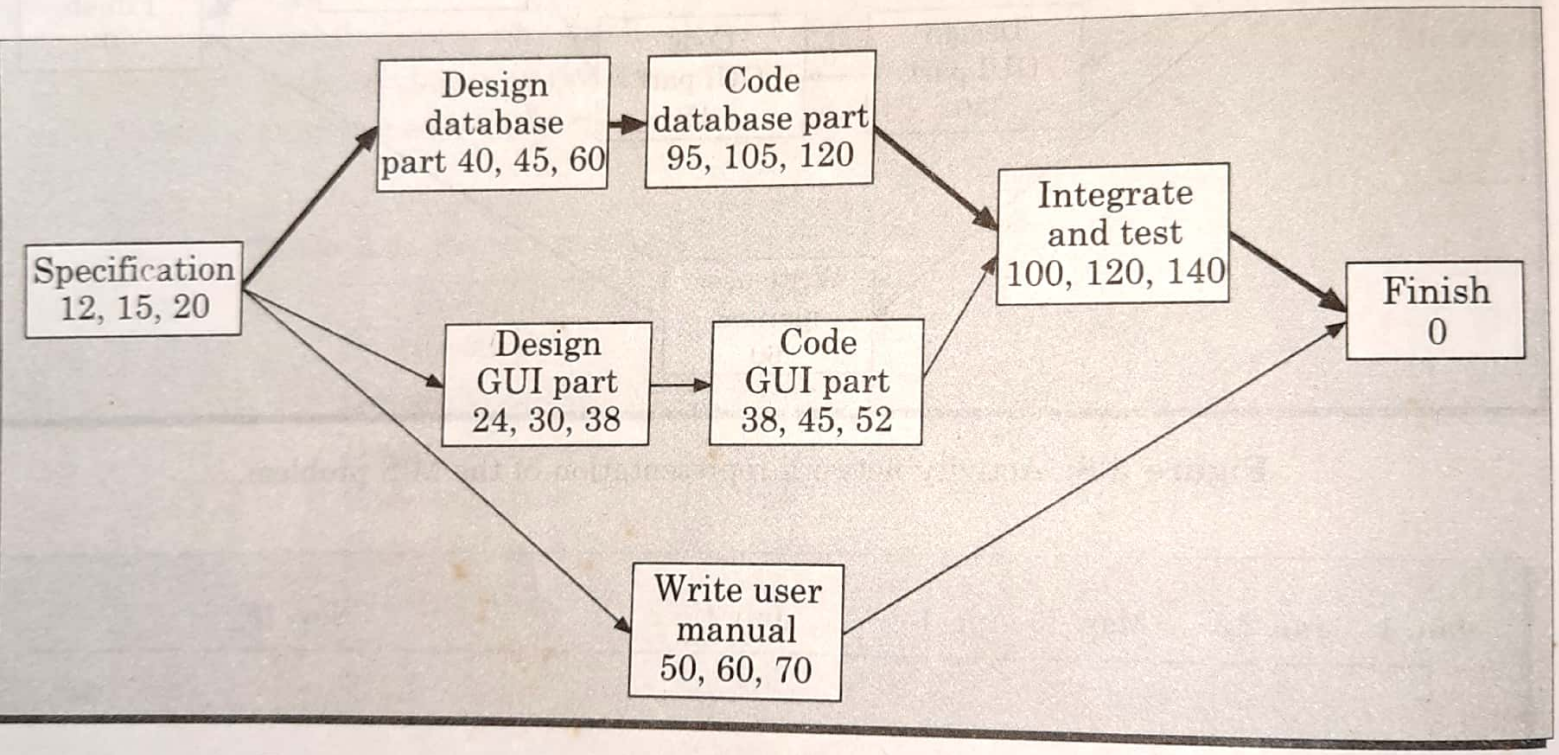


Figure 3.10: PERT chart representation of the MIS problem.

Figure 3.14: Mixed team structure.

3.11 STAFFING

Software project managers usually take the responsibility of choosing their team. Therefore, they need to identify good software developers for the success of the project. A common mis-

conception held by managers as evidenced in their staffing, planning and scheduling practices, is the assumption that one software engineer is as productive as another. However, experiments have revealed that there exists a large variability of productivity between the worst and the best software developers in a scale of 1 to 30. In fact, the worst developers may sometimes even reduce the overall productivity of the team, and thus in effect exhibit negative productivity. Therefore, choosing good software developers is crucial to the success of a project.

3.11.1 Who is a Good Software Engineer?

In the past, several studies concerning the traits of a good software engineer have been carried out. All these studies roughly agree on the following attributes that good software developers should possess:

1. Exposure to systematic techniques, i.e. familiarity with software engineering principles
2. Good technical knowledge of the project areas (*domain knowledge*)
3. Good programming abilities
4. Good communication skills like oral, written and interpersonal skills
5. High motivation
6. Sound knowledge of fundamentals of computer science
7. Intelligence
8. Ability to work in a team
9. Discipline

every activity. An approximate rule of thumb is to set a milestone

3.13 SOFTWARE CONFIGURATION MANAGEMENT

The results (also called as the deliverables) of a large software development effort typically consist of a large number of objects, e.g. source code, design document, SRS document, test document, user's manual, etc. These objects are usually referred to and modified by a number of software developers throughout the life cycle of the software. The state of all these objects at any point of time is called the *configuration* of the software product. The state of each deliverable object changes as development progresses and also as bugs are detected and fixed.

Software configuration management deals with effectively tracking and controlling the configuration of a software product during its life cycle.

Before we discuss configuration management, we must be clear about the distinction between a version and a revision of a software product. A new version of a software is created when there is significant change in functionality, technology, or the hardware it runs on, etc. On the other hand, a new release is created if there is only a bug fix, minor enhancements to the functionality, usability, etc. Even the initial delivery might consist of several versions and more versions might be added later on. For example, one version of a mathematical computation package might run on Unix-based machines, another on Microsoft Windows, and so on. As a software is released and used by the customer, errors are discovered that need correction. Enhancements to the functionalities of the software may also be needed. A new **release** of software is an improved system intended to replace an old one. Often systems are described as version m, release n; or simple m.n. Formally, a history relation is version of can be defined between objects. This relation can be split into two subrelations is revision of and is variant of. In the following, we first discuss the necessity of configuration management and subsequently we discuss the configuration management activities and tools.

3.13.1 Necessity of Software Configuration Management

There are several reasons for putting an object under configuration management. But, possibly the most important reason for configuration management is to control the access to the different deliverable objects. Unless strict discipline is enforced regarding updation and storage of different objects, several problems can appear. The following are some of the important problems that can appear if configuration management is not used.

Inconsistency problem when the objects are replicated

Consider a scenario where every software developer has a personal copy of an object (e.g. source code). As each developer makes changes to his local copy, he is expected to intimate the changes that he has made to other developer, so that the changes in interfaces are uniformly changed across all modules. However, many times a developer makes changes to the interfaces in his local copies and forgets to intimate other teammates about the changes. This makes the different copies of the object inconsistent. Finally, when the product is in-

tegrated, it does not work. Therefore, when several team members work on developing an object, it is necessary for them to work on a single copy of the object, otherwise inconsistency may arise.

Problems associated with concurrent access

Assume that only a single copy of a program module is maintained, and several developers are working on it. Two developers may simultaneously carry out changes to the different portions of the same module, and while saving overwrite each other. Though we explained the problem associated with concurrent access to program code, similar problems can occur for any other deliverable object.

Providing a stable development environment

When a project work is underway, the team members need a stable environment to make progress. Suppose one is trying to integrate module A, with the modules B and C. He cannot make progress if developer of module C keeps changing C; this can be especially frustrating if a change to module C forces recompilation of module A. When an effective configuration management is in place, the manager freezes the objects to form a *base line*. When any one needs to change any of the objects under configuration control, he is provided with a copy of the base line item. The requester makes changes to his private copy. Only after the requester is through with all modifications to his private copy, the configuration is updated and a new base line gets formed instantly. This establishes a baseline for others to use and depend on. Also, baselines may be archived periodically (Archiving means copying to a safe place such as a magnetic tape).

System accounting and maintaining status information

System accounting keeps track of who made a particular change to an object and when the change was made.

Handling variants

Existence of variants of a software product causes some peculiar problems. Suppose you have several variants of the same module, and find a bug in one of them. Then, it has to be fixed in all versions and revisions. To do it efficiently, you should not have to fix it in each and every version and revision of the software separately.

3.9.5 Project Monitoring and Control

Once the project gets underway, the project manager has to monitor the project continuously to ensure that it is progressing as per the plan. The project manager designates certain key events such as completion of some important activity as **milestones**. For example, a milestone can be the completion and review of the SRS document, completion of the coding and unit testing, etc. Once a milestone is reached, the project manager can assume that some measurable progress has been made. If any delay in reaching a milestone is predicted, then corrective actions might have to be taken. This may entail reworking all the schedules and producing a fresh schedule.

As already mentioned, the PERT charts are especially useful in project monitoring and control. A path in this graph is any set of consecutive nodes and edges in this graph from the starting node to the last node. A **critical path** in this graph is a path along which every milestone is critical to meeting the project deadline. In other words, if any delay occurs along a critical path, the entire project would get delayed. It is, therefore, necessary to identify all the critical paths in a schedule—adhering to the schedules of the tasks appearing on the critical paths is of prime importance to meet the delivery date. Please note that there may be

more than one critical path in a schedule. The tasks along a critical path are called *critical tasks*. If necessary, a manager may switch resources from a noncritical task to a critical task so that all milestones along the critical path are met.

Several tools are available which can help you to figure out the critical paths in an unrestricted schedule, but figuring out an optimal schedule with resource limitations and with a large number of parallel tasks is a very hard problem. There are several commercial products for automating the scheduling techniques are available. Popular tools to help draw the schedule-related graphs include the MS-Project software available on personal computers.