LECTURE # 1

1. Introduction & Fundamentals

1.1 What is Management?

Basically, the management involves the following activities:

- **Planning**- deciding what is to be done
- **Organizing**- making arrangements
- **Staffing**- selecting the right people for the job
- **Directing**- giving instructions
- **Monitoring**- checking on progress
- **Controlling**- taking action to remedy hold-ups
- **Innovating**- coming up with new solutions
- **Representing**- liaising with users, etc.

1.2 What is Project Management?

Project Management is the art of maximizing the probability that a project delivers its goals on **Time**, to **Budget** and at the required **Quality**.

The art of planning for the future has always been a human trait. In essence a project can be captured on paper with a few simple elements: a start date, an end date, the tasks that have to be carried out and when they should be finished, and some idea of the resources (people, machines etc) that will be needed during the course of the project.

Project management is the **application of knowledge, skills, tools, and techniques** to project activities to meet project requirements. Project management is accomplished through the use of the processes such as: initiating, planning, executing, controlling, and closing. It is important to note that many of the processes within project management are iterative in nature. This is in part due to the existence of and the necessity for progressive elaboration in a project throughout the project life cycle; i.e., the more you know about your project, the better you are able to manage it.

Project management is also defined as a **strategic competency** that has successfully been applied in such high profile projects as the construction of silk root, organizing and managing the Olympics Games, and the construction of Islamabad-Lahore motorway, just to name a few. If project management can play a major role in these success stories, just imagine what it might be able to do for your own organization.

The term project management is sometimes used to describe an organizational approach to the management of ongoing operations. This approach, more properly called **management by projects**, treats many aspects of ongoing operations as projects to apply project management techniques to them.

Almost any human activity that involves carrying out a non- repetitive task can be a project. So we are all project managers! We all practice project management (PM). But there is a big difference between carrying out a very simple project involving one or two people and one involving a complex mix of people, organizations and tasks.

1.3 What is Software Project Management?

When the plan starts to involve different things happening at different times, some of which are dependent on each other, plus resources required at different times and in different quantities and perhaps working at different rates, the paper plan could start to cover a vast area and be unreadable.

Nevertheless, the idea that complex plans could be analyzed by a computer to allow someone to control a project is the basis of much of the development in technology that now allows projects of any size and complexity, not only to be planned, but also modeled to answer 'what if?' questions.

The original programs and computers tended to produce answers long after an event had taken place. Now, there are many project planning and scheduling programs that can provide real time information, as well as linking to risk analysis, time recording, and costing, estimating and other aspects of project control.

But **computer programs are not project management: they are tools for project managers to use**. Project management is all that mix of components of control, leadership, teamwork, resource management etc that goes into a successful project.

Project managers can be found in all industries. Their numbers have grown rapidly as industry and commerce has realized that much of what it does is project work. And as project-based organizations have started to emerge, project management is becoming established as both a professional career path and a way of controlling business.

So opportunities in project management now exist not only in being a project manager, but also as part of the support team in a project or program office or as a team leader for part of a project. There are also qualifications that can be attained through the professional associations.

1.4 What is a Project?

A project is **an activity with specific goals which takes place over a finite period** of time.

"A temporary organization that is needed to produce a unique and pre-defined outcome or result at a pre-specified time using pre-determined resources"

Projects are often implemented as a means of achieving an organization's strategic plan. Operations and projects differ primarily in that operations are ongoing and repetitive while projects are temporary and unique. A project can thus be defined in terms of its distinctive characteristics—***a project is a temporary endeavor undertaken to create a unique product or servic*e. *Temporary*** means that every project has a definite beginning and a definite end. ***Unique*** means that the product or service is different in some distinguishing way from all other products or services. For many organizations, projects are a means to respond to those requests that cannot be addressed within the organization's normal operational limits.

Projects are undertaken at all levels of the organization. They may involve a single person or many thousands. Their duration ranges from a few weeks to more than five years. Projects may involve a single unit of one organization or may cross organizational boundaries, as in joint ventures and partnering.

Examples of projects include:

- Developing a new product or service.
- Effecting a change in structure, staffing, or style of an organization.
- Designing a new transportation vehicle.
- Developing or acquiring a new or modified information system.
- Constructing a building or facility.
- Building a water system for a community in a developing country.
- Running a campaign for political office.
- Implementing a new business procedure or process.

1. Temporary

Temporary means that every project has a definite beginning and a definite end. The end is reached when the project's objectives have been achieved, or it becomes clear that the project objectives will not or cannot be met, or the need for the project no longer exists and the project is terminated. Temporary does not necessarily mean short in duration; many projects last for several years. In every case, however, the duration of a project is finite; projects are not ongoing efforts.

2. Unique, Product Service or Result

Projects involve creating something that has not been done in exactly the same way before and which is, therefore, *unique* and distinct. Projects create:

- A product or artifact that is produced, is quantifiable and can be either an end item in itself or a component item
- A capability to perform a service, such as business functions supporting production or distribution
- A result, such as new knowledge. For example, a research and development project develops knowledge that can be used to determine whether or not a trend is present or a new process will benefit society.

The presence of repetitive elements does not change the fundamental uniqueness of the project work. For example:

- A project to develop a new commercial airliner may require multiple proto-types.
- A project to bring a new drug to market may require thousands of doses of the drug to support clinical trials.
- A real estate development project may include hundreds of individual units.
- A development project (e.g., water and sanitation) may be implemented in five geographic areas.

3. Aims/Tasks/Purpose

The projects are designed to achieve specific targets defined in terms of aims, tasks or a purpose. The nature and size of the project depends upon complexity of the task, realization of the aims and scope of the purpose any organization wants to achieve. In short project has to be aimed for achieving certain tasks in a given time frame.

4. Limited Time Scale

The projects are always designed considering time constraints. Extension to the project completion dead lines are always discouraged as time overrun, costs extra and in some cases opportunity cost for not completing a project is too high.

Progressive, Elaboration

Progressive elaboration is a characteristic of projects that accompanies the concepts of temporary and unique. "***Progressively***" means developing thoroughly in steps, and continuing steadily by increments while ***elaborated*** means "worked out with care and detail; developed thoroughly"

For example, the project scope will be broadly described early in the project, and made more explicit and detailed as the project team develops a better and more complete understanding of the objectives and deliverables.

Progressive elaboration of project specifications must be carefully coordinated with proper project scope definition, particularly if the project is performed under contract. When properly defined, the scope of the project—the work to be done—

should be controlled as the project and product specifications are progressively elaborated.

The following examples illustrate progressive elaboration in two different application areas.

- ✓ *Example 1*. Development of a chemical processing plant begins with process engineering to define the characteristics of the process. These characteristics are used to design the major processing units. This information becomes the basis for engineering design, which defines both the detail plant layout and the mechanical characteristics of the process units and ancillary facilities. All of this results in design drawings that are elaborated to produce fabrication and construction drawings. During construction, interpretations and adaptations are made as needed and subject to proper approval. This further elaboration of the deliverables is captured in as-built drawings, and final operating adjustments are made during testing and turnover.

- ✓ *Example 2*. The product of an economic development project may initially be defined as: "Improve the quality of life of the lowest income residents of community X." As the project proceeds, the products may be described more specifically as, for example: "Provide access to food and water to 500 low income residents in community X." The next round of progressive elaboration might focus exclusively on increasing agriculture production and marketing, with provision of water deemed to be a secondary priority to be initiated once the agricultural component is well under way.

LECTURE # 2

1. Introduction & Fundamentals

1.5 Goals of Project management

Project management is the discipline of defining and achieving a set of goals while optimizing the use of allocated resources (time, money, people, space, etc). This includes planning, scheduling and maintaining progress of the activities that comprise the project. Project management is normally reserved for focused, non-repetitive, time-limited activities with some degree of risk and that are beyond the usual scope of program (operational) activities for which the organization is responsible.

Project management software describes the tools to efficiently coordinate and automate the various project management component processes. Project management software generally offers extensive reporting features, such as day-to-day status updates of project progress, scheduling and dependency trees, and system-generated alerts when schedules slip beyond pre-set tolerances. Most project management tools include web-accessible interfaces so that employees can access features of the software relevant to their needs, and functionality to allow managers to share resource pools without overbooking.

1.6 Project Characteristics

1. Temporary
Temporary means that every project has a definite beginning and a definite end. The end is reached when the project's objectives have been achieved, or it becomes clear that the project objectives will not or cannot be met, or the need for the project no longer exists and the project is terminated. Temporary does not necessarily mean short in duration; many projects last for several years. In every case, however, the duration of a project is finite; projects are not ongoing efforts.

2. Unique, Product Service or Result
Projects involve creating something that has not been done in exactly the same way before and which is, therefore, *unique* and distinct. Projects create:

- A product or artifact that is produced, is quantifiable and can be either an end item in itself or a component item
- A capability to perform a service, such as business functions supporting production or distribution

- A result, such as new knowledge. For example, a research and development project develops knowledge that can be used to determine whether or not a trend is present or a new process will benefit society.

The presence of repetitive elements does not change the fundamental uniqueness of the project work. For example:

- A project to develop a new commercial airliner may require multiple proto-types.
- A project to bring a new drug to market may require thousands of doses of the drug to support clinical trials.
- A real estate development project may include hundreds of individual units.
- A development project (e.g., water and sanitation) may be implemented in five geographic areas.

5. Aims/Tasks/Purpose

The projects are designed to achieve specific targets defined in terms of aims, tasks or a purpose. The nature and size of the project depends upon complexity of the task, realization of the aims and scope of the purpose any organization wants to achieve. In short project has to be aimed for achieving certain tasks in a given time frame.

6. Limited Time Scale

The projects are always designed considering time constraints. Extension to the project completion dead lines are always discouraged as time overrun, costs extra and in some cases opportunity cost for not completing a project is too high.

1.7 Four Project Dimensions

Software project management is an umbrella activity within software engineering. It begins before any technical activity is initiated and continues throughout the definition, development, and support of computer software.

Four P's have a substantial influence on software project management- people, product, process, and project.

- **People** must be organized into **effective teams**, motivated to do high-quality software work, and coordinated to achieve effective communication.
- **The product requirements must be communicated from customer to developer,** partitioned (decomposed) into their constituent parts, and positioned for work by the software team.
- The **process** must be adapted to the people and the problem. A common process framework is selected, an appropriate software engineering paradigm is applied, and a set of work tasks is chosen to get the job done.

- **T**he **project** must be organized in a manner that enables the software team to succeed.

Effective software project management focuses on the four P's: people, product, process, and project. The order is not arbitrary. The manager who forgets that software engineering work is an intensely human endeavor will never have Success in project management. A manager who fails to encourage comprehensive customer communication early in the evolution of a project risks building an elegant solution for the wrong problem. The manager who pays little attention to the process runs the risk of inserting competent technical methods and tools into a vacuum. The manager who embarks without a solid project plan jeopardizes the success of the product.

⇒ **People**

In a study published by the IEEE, the engineering vice presidents of three major technology companies were asked the most important contributor to a successful software project. They answered in the following way:

VP 1: I guess if you had to pick one thing out that is most important in our environment. I'd say it's not the tools that we use, it's the people.
VP 2: The most important ingredient that was successful on this project was having smart people…very little else matters in my opinion....The most important thing you do for a project is selecting the staff...The success of the software development organization is very, very much associated with the ability to recruit good people.
VP 3: The only rule I have in management is to ensure I have good people – real good people-and that I grow good people – and that I provide an environment in which good people can produce.

Indeed, this is a compelling testimonial on the importance of people in the software engineering process. And yet, all of us, from senior engineering vice presidents to the lowliest practitioner, often take people for granted. Managers argue (as the preceding group had) that people are primary, but their actions sometimes belie their words. In this section we examine the players who participate in the software process and the manner in which they are organized to perform effective software engineering.

1. The Players

The software process (and every software project) is populated by players who can be categorized into one of five constituencies:

1. **Senior managers** who define the business issues that often have significant influence on the project.

2. **Project (technical) managers** who must plan, motivate, organize, and control the practitioners who do software work.
3. **Practitioners** who deliver the technical skills that are necessary to engineer a product or application.
4. **Customers** who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.

2. End-users

Who interact with the software once it is released for production use. Every software project is populated by people who fall within this taxonomy. To be effective, the project team must be organized in a way that maximizes each person's skills and abilities. And that's the job of the team leader.

3. Team Leaders

Project management is a people-intensive activity, and for this reason, competent practitioners often make poor team leaders. They simply don't have the right mix of people skills. And yet, as Edgemon states: "unfortunately and all too frequently it seems, individuals just fall into a project manager role and become accidental project managers." [EDG95].

The cultivation of motivated, highly skilled software people has been discussed since the 1960s (e.g., [COUBO] [WIT94} [DEM9B]). In fact, the "people factor' is so important that the Software Engineering Institute has developed a people management capability maturity model (PM-CMM), "to enhance the readiness of software organizations to undertake increasingly complex applications by helping to attract, grow, motivate, deploy, and retain the talent needed to improve their software development capability" [CUR94].

The people management maturity model defines the following key practice areas for software people: recruiting, selection, performance management, training, compensation, career development, organization and work design, and team/culture development. Organizations that achieve high levels of maturity in the people management area have a higher likelihood of implementing effective software engineering practices.

The PM-CMM is a companion to the software capability maturity model that guides organizations in the creation of a mature software process.

⇒ **The Process**

In a fascinating book that provides an economist's view of software and software engineering, Howard Baetjer. Jr, comments on the software process:

Software development is a social learning process. The process is a dialogue in which the knowledge that must become the software is brought together and embodied in the software. The process provides interaction between users and designers: between users and evolving tools, and between designers and evolving tools [technology] It is an iterative process in which the evolving tool itself serves as the medium for communication, with each new round of the dialogue eliciting more useful knowledge from the people involved.

When you build a product or system, it's important to go through a series of predictable steps – a road map that helps you create a timely, high-quality result, The road map that you follow is called a *'software process'* .

Software engineers and their managers adapt the process to their needs and then follow it. In addition, the people who have ties defined by the process requested the software play a role in the software process.

At a detailed level, the process that you adopt depends on the software you're building. One process might be appropriate for creating software for an aircraft avionics system, while an entirely different process would be indicated for the creation of a web site.

From the point of view of a software engineer, the work products are the programs, documents and data produces as a consequence of the software engineering activities defined by the process.

A software process provides the framework from which a comprehensive plan for software development can be established.

A small number of framework activities are applicable to all software projects, regardless of their size or complexity.

A number of different task sets-tasks, milestones, work products and; quality assurance points-enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team.

Finally, umbrella activities – such as software quality assurance, software configuration management, and measurement – overlay the process model. Umbrella activities are independent of anyone framework activity and occur throughout the process.

LECTURE # 3

1. Introduction & Fundamentals

1.8 Project Dimensions

⇒ **Product and Technology**

– **The 80:20, rule** was originated by Vilfredo Pareto, an Italian economist who studies the distribution of wealth in a variety of countries around 1900. He discovered a common phenomenon: about 80% of the wealth in most countries was controlled by a consistent minority -- about 20% of the people. Pareto called this a "predictable imbalance." His observation eventually became known as either the "80:20 rule" or "Pareto's Principle."

The credit for adapting Pareto's economic observations to business goes to the "Father of Total Quality Management," service quality consultant Joseph M. Juran. In 1950, he published "The Quality Control Handbook," which first recognized the applicability of the Pareto principle in the context of inventory management, e.g.:

- 20% of the repair parts normally account for 80 percent of the total inventory
- 80% of production volume usually comes from 20% of the producers

He subsequently recognized that this rule of thumb was universally applicable across fields of endeavor. As a credit to Pareto's work, Juran named his finding the Pareto Principle. This universal management theory became generalized as "the 80-20 Rule":

The "80:20 rule" has become one of the best known "leadership shorthand terms" reflecting the notion that most of the results (of a life, of a program, of a financial campaign) come from a minority of effort (or people, or input).

The Rule, states that a small number of causes (20%) is responsible for a large percentage (80%) of the effect. It means that in anything a few (20 percent) are vital and many (80 percent) are trivial.

There is an inherent imbalance between cause and effect, effort and reward, inputs and outputs, etc; and that imbalance tends to the ratio of 80:20. So, if we know which 20% of our work produces 80% of our income, we can do more of it and our income will increase proportionately!

You know 20 percent of you stock takes up 80 percent of your warehouse space and that 80 percent of your stock comes from 20 percent of your suppliers. Also 80 percent of your sales will come from 20 percent of your sales staff. 20 percent of your staff will cause 80 percent of your problems, but another 20 percent of your staff will provide 80 percent of your production. It works both ways.
Some Sample 80/20 Rule Applications

80% of process defects arise from 20% of the process issues.
20% of your sales force produces 80% of your company revenues.
80% of delays in schedule arise from 20% of the possible causes of the delays.
80% of customer complaints arise from 20% of your products or services.

How It Can Help You

- The value of the Pareto Principle for a manager is that it reminds you to focus on the 20 percent that matters. Of the things you do during your day, only 20 percent really matter. Those 20 percent produce 80 percent of your results. Identify and
Characteristic
focus on those things. When the fire drills of the day begin to sap your time, remind yourself of the 20 percent you need to focus on. If something in the schedule has to slip, if something isn't going to get done, make sure it's not part of that 20 percent.

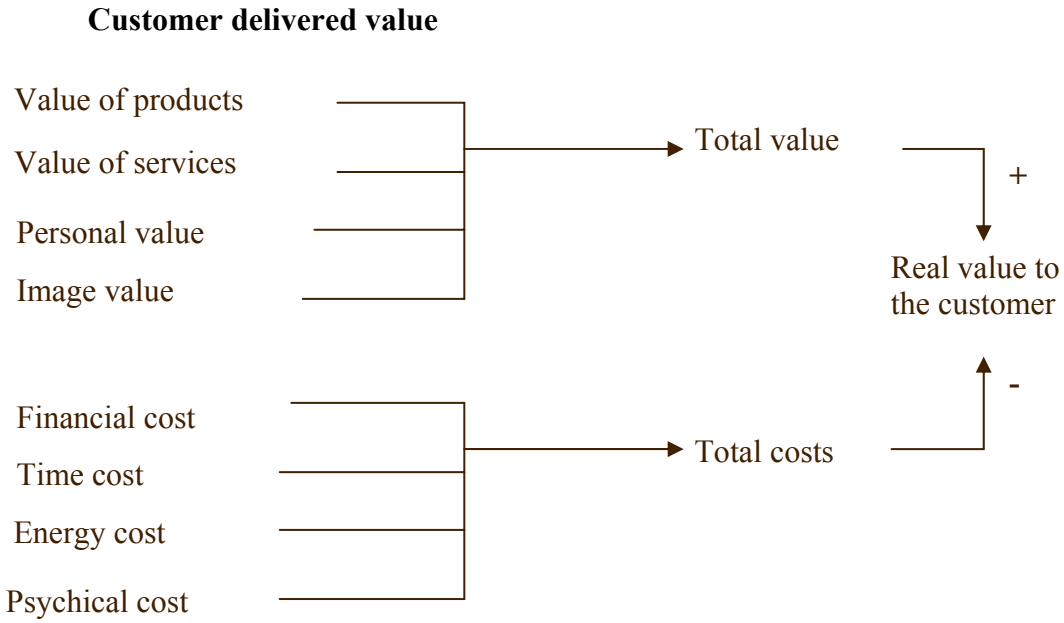
  Pareto's Principle, the 80/20 Rule, should serve as a daily reminder to focus 80 percent of your time and energy on the 20 percent of you work that is really important. Don't just "work smart", work smart on the right things.

- **Size**

  The larger product, there will be more requirements and features to deliver, eventually it will take more time in its production. So if you cut the size of the produce to half it will save you 60% of the effort.

- **Characteristic**

- **Development Tools**



1.9 Project Phases

Organizations performing projects will usually divide each project into several ***Project phases*** to improve management control and provide for links to the ongoing operations of the performing organization.

Collectively, the project phases are known as the ***project life cycle***. Software development, just like most other activities, has a beginning, middle and an end.

The end of one development activity is sometimes perceived as being linked to the beginning of a new development activity thus producing a cycle of beginning-middle-end, link, beginning-middle-end, link, and so forth.

This view of software development is referred to as the **software development life cycle**.

A project has five phases. Here's a brief summary of each:

- ⇒ **Initiation**
  Articulate your vision for the project, establish goals, assemble your team, and define expectations and the scope of your project.
- ⇒ **Planning**
  Refine the scope, identify specific tasks and activities to be completed, and develop a schedule and budget.
- ⇒ **Executing**
  Accomplish your goals by leading your team, solving problems, and building your project.

⇒ **Controlling**
Monitor changes to the project make corrections, adjust your schedule to respond to problems, or adjust your expectations and goals.

⇒ **Closing**
Deliver your project to your audience, acknowledge results, and assess its success. Take the time to compose a written evaluation of the project and the development effort.

The middle three phases are not sequential. You will find that you are constantly planning, executing, and controlling your project as necessary.

Aren't these phases really just common sense? In many ways, yes, but keep in mind that software development, whether a few Web pages or a complex CD-ROM, is a complex, unpredictable process.

Most software projects (something like 80 percent) are delivered late, substantially over budget, and incomplete. The more effort you put into managing your project, the more you increase your chances of success.

- **Characteristics of Project Phases**

Each project phase is marked by completion of one or more deliverables. A *deliverable* is a tangible, verifiable work product such as a feasibility study, a detail design, or a working prototype. The deliverables, and hence the phases, are part of a generally sequential logic designed to ensure proper definition of the product of the project.

The conclusion of a project phase is generally marked by a review of both key deliverables and project performance to date, to a) determine if the project should continue into its next phase and b) detect and correct errors cost effectively. These phase-end reviews are often called *phase exits*, *stage gates*, or *kill points*.

Each project phase normally includes a set of defined deliverables designed to establish the desired level of management control. The majority of these items are related to the primary phase deliverable, and the phases typically take their names from these items: requirements, design, build, test, startup, turnover, and others, as appropriate.

- **Characteristics of the Project Life Cycle**

The project life cycle serves to define the beginning and the end of a project. For example, when an organization identifies an opportunity to which it would like to respond, it will often authorize a needs assessment and/or a feasibility study to decide if it should undertake a project. The project life-cycle definition will determine whether the feasibility study is treated as the first project phase or as a separate, standalone project.

The project life-cycle definition will also determine which transitional actions at the beginning and the end of the project are included and which are not. In this manner, the project life-cycle definition can be used to link the project to the ongoing operations of the performing organization.

The phase sequence defined by most project life cycles generally involves some form of technology transfer or handoff such as requirements to design, construction to operations, or design to manufacturing. Deliverables from the preceding phase are usually approved before work starts on the next phase. However, a subsequent phase is sometimes begun prior to approval of the previous phase deliverables when the risks involved are deemed acceptable. This practice of overlapping phases is often called ***fast tracking***.

Project life cycles generally define:

- What technical work should be done in each phase (e.g., is the work of the architect part of the definition phase or part of the execution phase?).
- Who should be involved in each phase (e.g., implementers who need to be involved with requirements and design).
- Project life-cycle descriptions may be very general or very detailed. Highly detailed descriptions may have numerous forms, charts, and checklists to provide structure and consistency. Such detailed approaches are often called *project management methodologies*.

- Most project life-cycle descriptions share a number of common characteristics:

- Cost and staffing levels are low at the start, higher toward the end, and drop rapidly as the project draws to a conclusion.
- The probability of successfully completing the project is lowest, and hence risk and uncertainty are highest, at the start of the project. The probability of successful completion generally gets progressively higher as the project continues.
- The ability of the stakeholders to influence the final characteristics of the project's product and the final cost of the project is highest at the start and gets progressively lower as the project continues. A major contributor to this phenomenon is that the cost of changes and error correction generally increases as the project continues.

Care should be taken to distinguish the *project* life cycle from the *product* life cycle. For example, a project undertaken to bring a new desktop computer to market is but one phase or stage of the product life cycle.

Although many project life cycles have similar phase names with similar deliverables required, few are identical. Most have four or five phases, but some

have nine or more. Even within a single application area, there can be significant variations.

One organization's software development life cycle may have a single design phase while another's has separate phases for functional and detail design.

Subprojects within projects may also have distinct project life cycles. For example, an architectural firm hired to design a new office building is first involved in the owner's definition phase when doing the design, and in the owner's implementation phase when supporting the construction effort. The architect's design project, however, will have its own series of phases from conceptual development through definition and implementation to closure. The architect may even treat designing the facility and supporting the construction as separate projects with their own distinct phases.

- **Project Life Cycle includes the following Phases and activities:**

A. Concept Phase

1. User Need
2. Initial Investigation
3. User Review
4. System Performance Design
5. Candidate Review
6. Study Phase Report

B. Requirements Phase

1. The software requirements specification document
2. The project development plan
3. The software test plan

C. Design Phase

1. General System Review
2. Processing Requirements Identification
3. Data Base Design
4. Control Requirements
5. Output Design
6. Input Design
7. Software Selection
8. Equipment Selection/Acquisition
9. People
10. Reference Manual Identification
11. Plans
12. Design Specifications Preparation
13. Design Phase Report Preparation

D. Development Phase

1. Implementation Planning
2. Computer Program Design
3. User Review
4. Equipment Acquisition and Installation
5. Coding and Debugging
6. Computer Program Testing
7. System Testing
8. Reference Manual Preparation
9. Personnel Training
10. Changeover Plan Preparation
11. Development Phase Report Preparation
12. User Acceptance Review

E. Operation Phase

1. System Changeover
2. Routine Operation
3. System Performance Evaluation
4. System Changes/Enhancements

1.10 Software Development Lifecycle

⇒ **Water Fall Theme**

Software development, just like most other activities, has a beginning, middle and an end. The end of one development activity is sometimes perceived as being linked to the beginning of a new development activity thus producing a cycle of beginning-middle-end, link, beginning-middle-end, link, and so forth. This view of software development is referred to as the software development life cycle.

There are many variations of the software development life cycle. Figure 1 presents a simple life cycle that was common during the first few decades of software development. In those early days of software development, the programmer would create programs by iterating from code to fix then back to code, and then to fix again, until something acceptable was (hopefully) produced. At the start of the cycle, there was usually no clear concept of what was required, and the basic development procedure was a form of 'let's see what we can do' approach.

The software development method represented by the development cycle in Fig.1 is often referred to as the code and fix method (for obvious reasons). Software development methodologies have come a long way since the days of code and fix, though it is surprising how much software is still being developed this way.

Successful management of any project, especially software projects, requires planning, and planning is impossible with code and fix, which is totally unpredictable. Management of software development within an engineering

discipline is based on a much more orderly set of development phases. These phases are not implemented solely by programmers; they require software engineers. In fact, programming has become a relatively small part of the modern software development cycle, as is evident from Table 1.

The numbers in Table 1 are derived from the general shift in emphasis to software planning (requirements and design) and testing. Commercial data processing systems, with some exceptions, still spend a significant amount of development time in the programming and unit testing phase. Real-time systems are often more complex, and may include extensive hardware software integration. This usually requires more planning and more integration and testing.

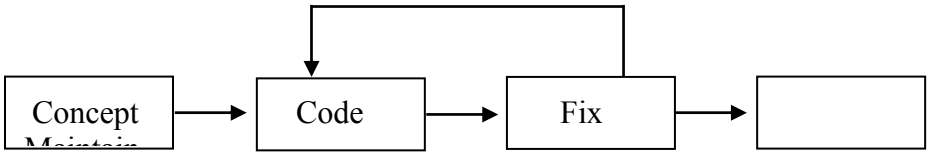


Figure 1: the code and fix method

Table 1 Estimated percentage of time spent in each major software development phase

	Planning	**Code and unit test**	**Integration and test**
Commercial data processing	25%	40%	35%
Real-time systems	35%	25%	40%
Military systems	40%	20%	40%

Military systems require high reliability and are usually closely supervised by the customer, leading to a significant increase in the time spent in planning.

The data in Table 1, of course, represents a generalization; commercial data processing systems can be just as complex as a real-time system.

Figure 2 presents the basic phased model of a software development cycle. This model, called the Waterfall model, gets its name from the way in which each phase cascades into the next (due to overlapping), as demonstrated in Fig. 3. Some interpretations of the Waterfall model, like the one that follows, combine the top level design and the detailed design phases into a single design phase, and the integration and test phases into a single phase. In fact, there are many variations of the classic Waterfall model, but they are all based upon a systematic transition from one development phase to the next, until the project is complete.

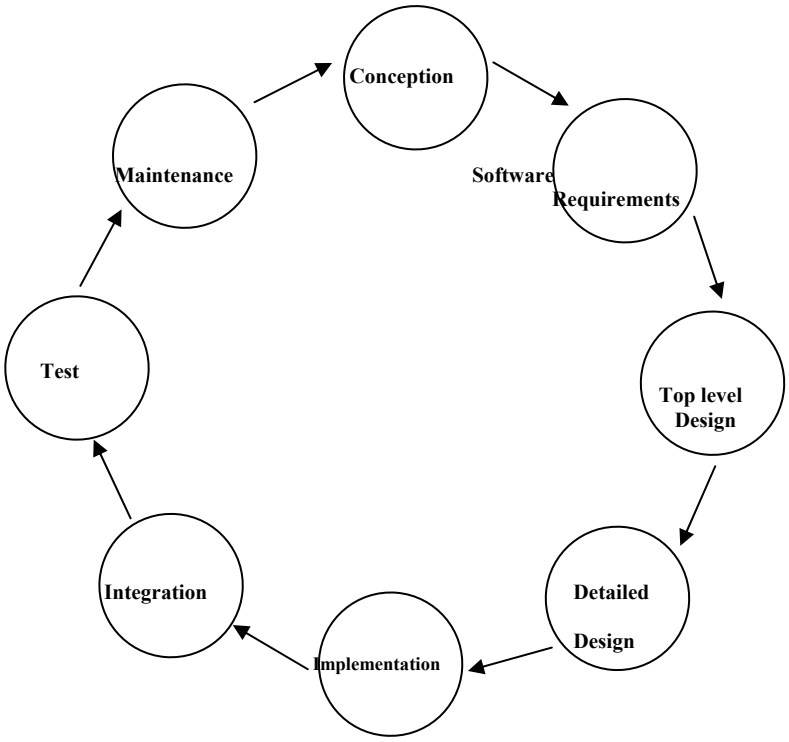


Figure 2: The phased model of the software development life cycle

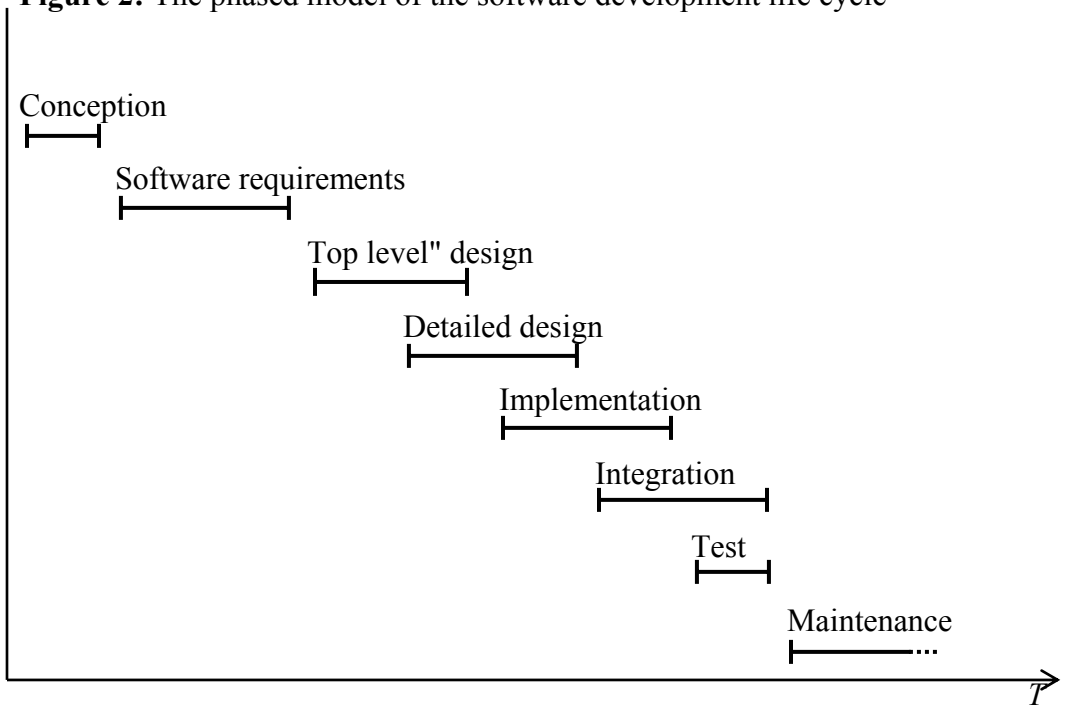


Figure 3: The Waterfall model of the software development life cycle

⇒ **Rapid prototyping** There are other development methodologies that do not move from one phase to the next like the Waterfall model. Rapid prototyping, for

instance, iterates in a mini-development phase until a system prototype is developed (see Fig. 4). After the prototype is complete, the Waterfall approach can then be implemented to complete the full system. Rapid prototyping is particularly helpful in projects where the requirements are difficult to specify. The prototype can be used as a tool for analyzing and determining what the requirements should be.

⇒ **The Spiral model**, described by Boehm (1988), is another development method that iterates between the requirements, design and implementation phases. However, the Spiral model continues iterating until the final system is complete. Within each, iteration, the Spiral model follows a phased approach similar to the Waterfall model.

Different models maybe suitable for different software projects or for different software development organizations However, a good model must include certain fundamental features. Some of these basic requirements are discussed in IEEE Standard (IEEE 1993) Standard for Software Life Cycle Processes. This standard describes the processes that are mandatory for the development of software and specifies the activities that must be included in the life cycle model.

Most modern software development models, and certainly those following IEEE Standard 1074, include some form of the basic phased model. It is therefore important to understand the different phases and how they relate to one another.

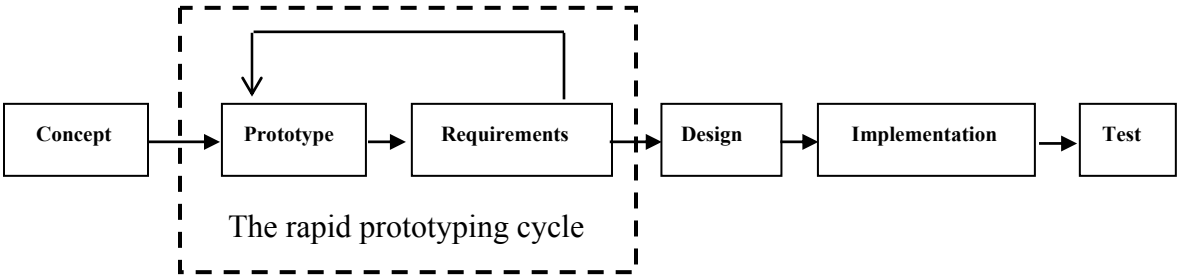


Figure 4: Rapid prototyping followed by the phase method.

LECTURE # 4

1. Introduction & Fundamentals

1.11 Costs and Cost Management

Project Cost Management includes the processes required to ensure that the project is completed within the approved budget**.**

- ⇒ **Resource Planning**—determining what resources (people, equipment, materials and what quantities of each should be used to perform project activities.
- ⇒ **Cost Estimating**—developing an approximation (estimate) of the costs of the resources needed to complete project activities.
- ⇒ **Cost Budgeting**—allocating the overall cost estimate to individual work activities.
- ⇒ **Cost Control**—controlling changes to the project budget.

These processes interact with each other and with the processes in the other knowledge areas as well.

Each process may involve effort from one or more individuals or groups of individuals, based on the needs of the project.

Each process generally occurs at least once in every project phase.

Although the processes are presented here as discrete elements with well-defined interfaces, in practice they may overlap and interact in ways not detailed here.

Project cost management is primarily concerned with the cost of the resources needed to complete project activities.

However, project cost management should also consider the effect of project decisions on the cost of using the project's product.

For example, limiting the number of design reviews may reduce the cost of the project at the expense of an increase in the customer's operating costs. This broader view of project cost management is often called life-cycle costing. Life-cycle costing together with Value Engineering techniques are used to reduce cost and time, improve quality and performance, and optimize the decision-making.

In many application areas, predicting and analyzing the prospective financial performance of the project's product is done outside the project.

In others (e.g., capital facilities projects), project cost management also includes this work. When such predictions and analyses are included, project cost management will include additional processes and numerous general management techniques such as return on investment, discounted cash flow, payback analysis, and others.

Project cost management should consider the information needs of the project stakeholders—different stakeholders may measure project costs in different ways and at different times. For example, the cost of a procurement item may be measured when committed, ordered, delivered, incurred, or recorded for accounting purposes.

1.12 Project vs. Program Management

⇒ **What is a Program?**

"A coordinated portfolio of projects that change organizations to achieve benefits of strategic importance"

Program management is the process of managing multiple on going projects. An example would be that of designing, manufacturing and providing support infrastructure for an automobile make.

It can be argued that Program Management has evolved from the complexities of the more intricate aspects of Project Management.

As projects became larger; more interrelated; complex and multidimensional, the need arose to have an approach that controlled Project Management whilst remaining focused on the strategic objectives of the business. Whilst Project Management focused on technical delivery, Program Management engaged on relating design concepts to the business strategic vision of the future.

It is very important that you understand the concept of Program Management as a method. Approaches may vary but definitions are relatively common. It is appropriate to understand the 'mission, goals and objectives' of Program Management and then relate these to the Program you propose to develop. Definitions will fall within each of these headings and it is possible to develop your own template which fits your organizational program. For example:

- **Mission:**

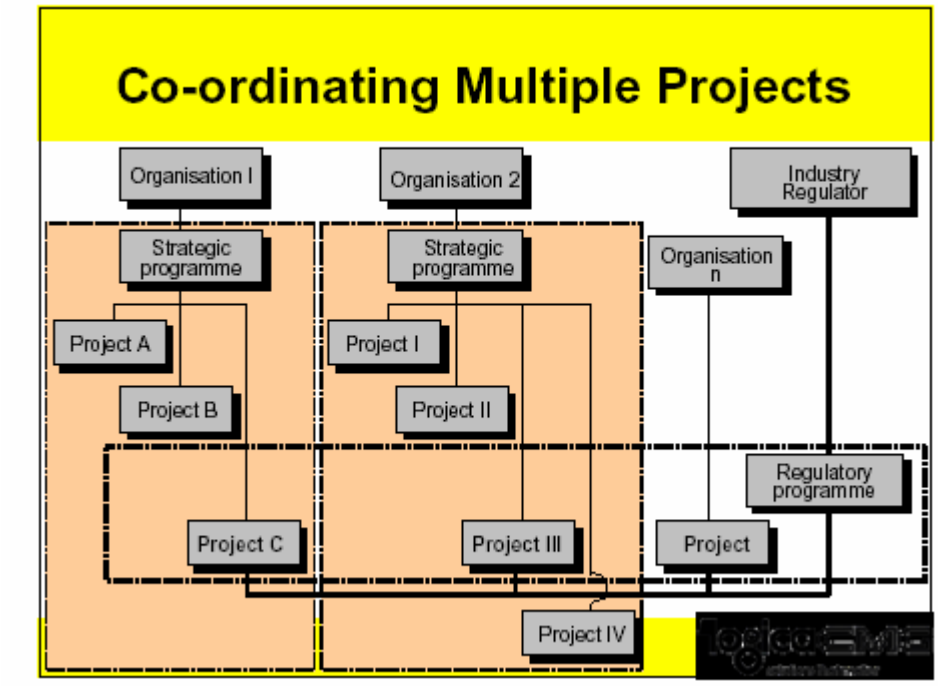
  To co-ordinate a portfolio of projects to harmonize communications in order to achieve a set of stated business objectives: *Provision of strategy alignment, with design objectives, in order to maintain control over a*

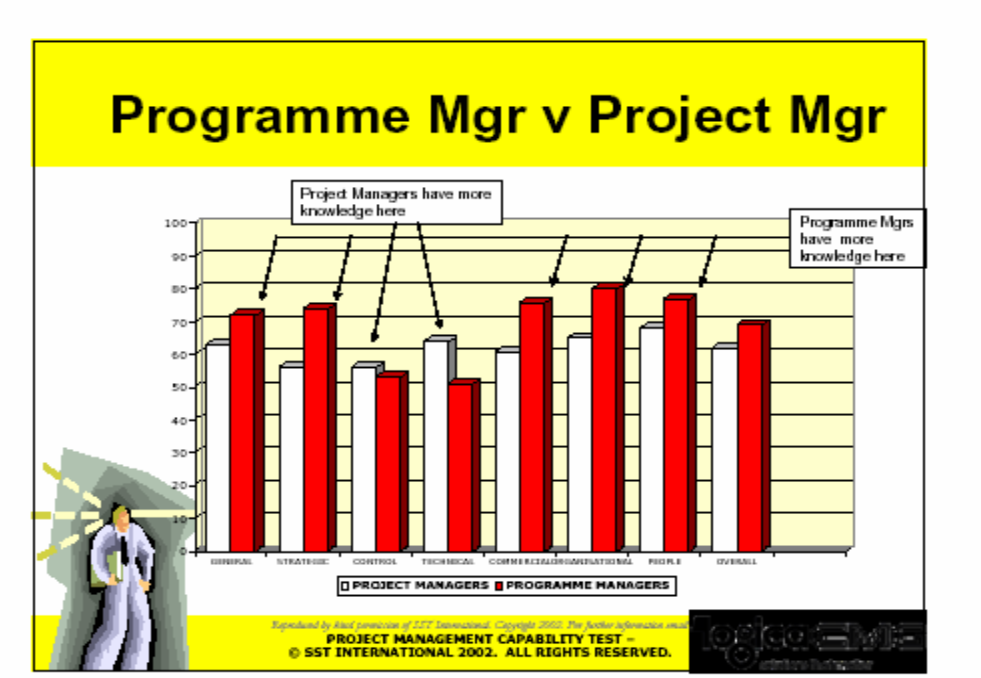
multiple project environment; ensuring quality end deliverables which meet business operational needs.

The above is a fairly complex mission statement but provides a framework to develop an intricate set of goals in order to utilize Program Management. It is possible to develop an extensive list of goals, depending upon the level of detail you wish to acquire. The following are offered as suggestions:

– **Goals:**

1. Clearly defined roles and responsibilities
2. Established baselines and Terms of Reference statement
3. Type of program defined
4. Future business blueprint
5. Recognition of business transformation procedures
6. Defined structure of the Program
7. Route Map
8. Visible end deliverables — vision of the future
9. Identification of future benefits
10. Risk
11. Contingency planning





Each of the goals would then be analyzed for providing objectives. For example:

- Managing a program embraces functions, risks and strategies outside of what a project manager does.
- A program has goals beyond those of a project or group of projects. Program outcomes are usually service delivery focused whereas a project is more likely to be focused on the delivery of a 'product'.
- A program is more than a grouping of projects (that's just a program of works).
- The coordinated management of a portfolio of projects to achieve a set of business objectives
- There are many more meanings of the term program management. Here are the more common meanings:

  – **The Multi-Project Organization:**

  Program Management is the directing of a portfolio of projects that benefit from a consolidated approach.

  Jobbing engineering companies; software houses contracting for work; and many other types of organization; run many simultaneous projects each of which may or not contribute towards the corporate goals.

  Typically the result of such a project is a deliverable which is eventually delivered to a client for payment. After many delays the payment arrives and

gets paid into the company's bank account thereby increasing cash flow which is achieving one of the company's objectives.

Sometimes the projects are much more directly aimed at corporate goals - opening a new factory or launching a new product - spring to mind.

The common elements of the projects are that they run simultaneously or at least overlap with each other, they share resources and are supposed to generate some income. One project being cancelled does not necessarily change the organization's general direction. These types of programs run for ever and need have no end date. The projects are separate in that there need not be logical links between projects. Whilst they share the same resources, delays in one project need not cause delays in others.

- **The Mega Project**

The management of a portfolio of projects towards one specific objective; Program management can also mean one very large project.

The USA's Man on the Moon Project was such a program. In this sense the term program indicates one very large project which is made up from a number of components. Within the Apollo program there were many projects: the Lunar Lander, the Orbiter, the Launcher and the Control Systems were all projects which were large, complex and interesting. Polaris and the Manhattan project (which resulted in the nuclear bomb) are other famous projects large enough to be called programs. Therefore, particularly in USA, the word program refers to a series of projects which make up one large project.

The program is usually reflected in the management structure as there will be a program manager to whom the project managers will report. Said program manager or sometimes program director will concern himself with recruiting and maintaining his project management teams and on integrating the deliverables of each project into one overall program. In this meaning of program management there is likely to be one physical deliverable.

These sorts of programs end. There will be a time when the overall objective has been achieved and the program and all of its constituent projects are over.

The projects within this type of program are often linked. Delays with one project often cause knock on effects with others due to logical links between tasks in both projects.

For example if the moon rocket launch pad project was delayed, it would delay the testing of the moon rocket itself. The Beirut Shopping Mall will be of little use without the water treatment plant and the new sewer scheme. Such

projects may not share the same resources but there are almost certain to be linked through their logic.

1.13 Project Success

Project success is correlated with thorough analyses of the need for project deliverables. Our research has shown that when a project results in deliverables that are designed to meet a thoroughly documented need, then there is a greater likelihood of project success. So managers should insist that there is a documented business need for the project before they agree to consume organizational resources in completing it.

We conduct planned and controlled software projects for one primary reason - it is the only known way to manage complexity. And yet, we still struggle. In 1998, industry data indicated that 26 percent of software projects failed outright and 46 percent experienced cost and schedule overruns [REE99].

Although the success rate for software projects has improved somewhat, our project failure rate remains higher than it should be.

In order to avoid project failure, a software project manager and the software engineers who build the product must avoid a set of common warning signs, understand the critical success factors that lead to good project management, and develop a common sense approach for planning, monitoring and controlling the project.

In order to manage a successful software project, we must understand what can go wrong (so that problems can be avoided) and how to do it right. In an excellent paper on software projects, John Reel [REE99] defines ten signs that indicate that an information systems project is in jeopardy:

1. Software people don't understand their customer's needs.
2. The product scope is poorly defined.
3. Changes are managed poorly.
4. The chosen technology changes.
5. Business needs change (or are ill-defined)
6. Deadlines are unrealistic.
7. Users are resistant.
8. Sponsorship is lost [or was never properly obtained).
9. The project team lacks people with appropriate skills.
10. Managers [and practitioners) avoid best practices and lessons learned.

Jaded industry professionals often refer to the 90-90 rule when discussing particularly difficult software projects: The first 90 percent of a system absorbs 90 percent of the allotted effort and time. The last 10 percent takes the other 90

percent of the allotted effort and time [ZAH94]. The seeds that lead to the 90-90 rule are contained in the signs noted in the preceding list.

But enough negativity! How does a manager act to avoid the problems just noted? Reel [REE99] suggests a five-part commonsense approach to software projects:

1. **Start on the right foot**. This is accomplished by working hard (very hard) to understand the problem that is to be solved and then setting realistic objects and expectations for everyone who will be involved in the project. It is reinforced by building the right team (Section 3.2.3) and giving the team the autonomy, authority, and technology needed to do the job.
2. **Maintain momentum**. Many projects get off to a good start and then slowly disintegrate. To maintain momentum, the project manager must provide incentives to keep turnover of personnel to an absolute minimum, the team should emphasize quality in every task it performs, and senior management should do everything possible to stay out of (the team's way.
3. **Track progress.** For a software project, progress is tracked as work products (e.g., specifications, source code, sets of test cases) are produced and approved (using formal technical reviews) as part of a quality assurance activation, software process and project measures can be collected and used to assess progress against averages developed for the software development organization.
4. **Make smart decisions**. In essence, the decisions *of* the project manager and the software team should be to "keep it simple." Whenever possible, decide to use commercial off-the-shelf software or existing software components, decide to avoid custom interfaces when standard approaches are available, decide to identify and then avoid obvious risks, and decide to allocate more time than you think is needed to complex or risky tasks (you'll need every minute).
5. **Conduct a postmortem analysis.** Establish a consistent mechanism for extracting lessons learned for each project. Evaluate the planned and actual schedules, collect and analyze software project metrics, get feedback from team members and customers, and record findings in written form.

– **Projects and Strategy**

Projects are a means of organizing activities that cannot be addressed within the organization's normal operational limits.

Projects are therefore often utilized as a means of achieving an organization's strategic plan, whether the project team is employed by the organization or is a contracted service provider**.**

Projects are typically authorized as a result of one or more of the following strategic considerations:

- A market demand (e.g., an oil company authorizes a project to build a new refinery in response to chronic gasoline shortages)
- A business need (e.g., a training company authorizes a project to create a new course in order to increase its revenues)
- A customer request (e.g., an electric utility authorizes a project to build a new substation to serve a new industrial park)
- A technological advance (e.g., an electronics firm authorizes a new project to develop a new generation of video game player after the introduction of the corresponding new game format)
- A legal requirement (e.g., a paint manufacturer authorizes a project to establish guidelines for the handling of a new toxic material).

No project ever goes 100% as planned, so project managers must learn to adapt to change. There are many things that can go wrong with project management. These are commonly called barriers. **Here are some possible barriers**:

1. **Poor Communication**

   - Many times a project may fail because the project team does not know exactly what to get done or what's already been done.

2. **Disagreement**

   - Project must meet all elements in a contract.
   - Customer and project manager must agree on numerous elements.
   - Failure to comply with standards and regulations.
   - Inclement weather.
   - Union strikes.
   - Personality conflicts.
   - Poor management

3. **Poorly defined project goals**

1.14 Trade-Off Triangle

Good project management deals with three factors: time, cost and performance.

Projects are successful if they are completed on time, within budget, and to performance requirements. In order to bring the many components of a large project into control there is a large toolkit of techniques, methodologies, and tools.

These techniques provide the tools for managing different components involved in a project: planning and scheduling, developing a product; managing financial and capital resources, and monitoring progress. However the success of a project will always rest on the abilities of a project manager and the team members.

In managing competing project requirements Project managers often talk of a triple constraint:

- Project scope
- Time and
- Cost

Project quality is affected by balancing these three factors.

High quality projects deliver the required product or service within scope, on time and within budget.

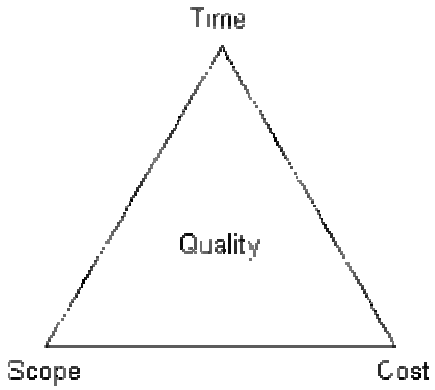
The relationship among these factors is such that if any one of the three factors changes, at least one other factor must change.

Simply put: project success means completing all project deliverables on *time,* within *budget,* and to a level of *quality* that is acceptable to sponsors and stakeholders.

The project manager must keep the team's attention focused on achieving these broad goals. Most people still want their projects to be on time, meet quality objectives, and not cost more than the budget. These form the classic time, quality, cost triangle.

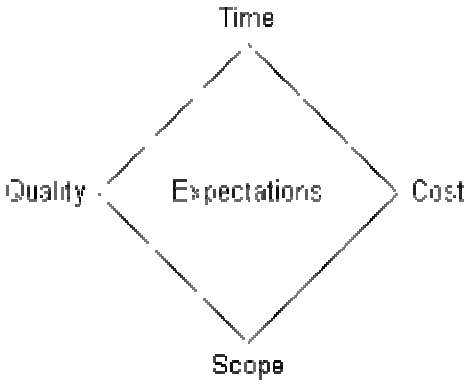
In fact if you have an unlimited budget and unlimited time, project management becomes rather easy. For most people, however, time and money are critical and that is what makes project management so important today.

Project management is often summarized in a triangle. The three most important factors are time, cost and scope. These form the vertices with quality as a central theme.



1. Projects must be delivered on time.
2. Projects must be within cost
3. Projects must be within scope
4. Projects must meet customer quality requirements

More recently, this has given way to a project management diamond, with time, cost, scope and quality the four vertices and customer expectations as a central theme. No two customers' expectations are the same so you must ask what their expectations are.



A project goes through four phases during its life:

1. Project Definition: Defining the goals, objectives and critical success factors for the project
2. Project Initiation: Everything that is needed to set-up the project before work can start
3. Project Control: Ensuring that a project stays on track and taking appropriate action to ensure it does
4. Project Closure: Disbanding of all the elements that were required to run the project

1.15 Project Management Skills

The role of the Leader in project management is one of great responsibility. It's the project manager's job to direct and supervise the project from beginning to end. Here are some other roles:

(a) Leadership
(b) Communications
(c) Problem Solving
(d) Negotiating
(e) Influencing the Organization
(f) Mentoring
(g) Process and technical expertise

(a) Leadership

Leadership is a complex phenomenon involving the leader, the followers, and the situation. Perhaps the best way for you to begin to understand the complexities of leadership is to see some of the ways leadership has been defined. Leadership researchers have defined leadership in the following different ways:

- *The creative and directive force of morale. (*Munson, 1921)
- *The process by which an agent induces a subordinate to behave in a desired manner.* (Bennis, 1959)
- *The presence of a particular influence relationship between two or more persons.* (Hollander & Julian, 1969)
- *Directing and coordinating the work of group members.* (Fiedler, 1967) *An interpersonal relation in which others comply because they want to, not because they have to*. (Merton, 1969; Hogan, Curphy, & Hogan, 1994)
- *Transforming followers, creating visions of the goals that may be attained, and articulating for the followers the ways to attain those goals.* (Bass, 1985; Fichy & Devanna, 1986)
- *The process of influencing an organized group toward accomplishing its goals.* (Roach & Behling, 1984)
- *Actions that focus resources to create desirable opportunities.* (Campbell, 1991)

- *The leader's job is to create conditions for the team to be effective.* (Ginnett, 1996)

Leading and managing are both essential management skills: one without the other is likely to produce poor results. Managing is primarily concerned with "consistently producing key results expected by stakeholders," while leading involves:

- Establishing direction—developing both a vision of the future and strategies for producing the changes needed to achieve that vision.
- Aligning people—communicating the vision by words and deeds to all those, whose cooperation may be needed to achieve the vision.
- Motivating and inspiring—helping people energize themselves to overcome political, bureaucratic, and resource barriers to change.

On a project, particularly a larger project, the project manager is generally expected to be the project's leader as well.

Leadership is not, however, limited to the project manager: it may be demonstrated by many different individuals at many different times during the project.

Leadership must be demonstrated at all levels of the project (project leadership, technical leadership, and team leadership).

(b) Communicating

Communicating involves the exchange of information and the ability to transmit and receive information with a high probability that the intended message is passed from sender to receiver.

The sender is responsible for making the information clear, unambiguous, and complete so that the receiver can receive it correctly. The receiver is responsible for making sure that the information is received in its entirety and understood correctly.

Few skills are more vital to leadership. Studies show that good leaders communicate feelings and ideas, actively solicit new ideas from others, and effectively articulate arguments, advocate positions, and persuade others.

The quality of a Leader's communication is positively correlated with subordinate satisfaction as well as with productivity and quality of services rendered.

Effective communication skills are also important because they provide leaders and followers with greater access to information relevant to important organizational decisions.

Communicating has many dimensions:

- *Written and oral, listening and speaking*
- *Internal* (within the project) and external (to the customer, the media, the public, etc.).
- *Formal* (reports, briefings, etc.) *and informal* (memos, ad hoc *conversations*, etc.)
- *Vertical* (up and down the organization) *and horizontal* (with peers and partner organization)

The general management skill of communicating is related to, but not the same as, Project Communications Management.

Communicating is the broader subject and involves a substantial body of knowledge that is not unique to the project context, for example:

- **Sender-receiver models**—feedback loops, barriers to communications, etc.
- **Choice of media**—when to communicate in writing, when to communicate orally, when to write an informal memo, when to write a formal report, etc.
- **Writing style**—active versus passive voice, sentence structure, word choice, etc.
- **Presentation techniques**—body language, design of visual aids, etc.
- **Meeting management techniques**—preparing an agenda, dealing with conflict, etc.

Project Communications Management is the application of these broad concepts to the specific needs of a project—for example, deciding how; when; in what form; and to whom to report project performance.

(c) Negotiating

Negotiating involves conferring with others to come to terms with them or reach an agreement.

Agreements may be negotiated directly or with assistance; mediation and arbitration are two types of assisted negotiation.

Negotiation is an approach that may help resolve some conflicts. Some important tips on negotiation include; preparing for a negotiation session;

keeping people and problems separate; focusing on issues, not positions and seeking win-win outcomes.

Negotiations occur around many issues, at many times, and at many levels of the project.

During the course of a typical project, project staff is likely to negotiate for any or all of the following:

- Scope, cost, and schedule objectives.
- Changes to scope, cost, or schedule.
- Contract terms and conditions.
- Assignments.
- Resources.

(d) Problem Solving

There are three steps involved in this important leadership role; identifying problem; analyzing its cause; and solving the problem.

Problem solving involves a combination of problem definition and decision-making.

Problem definition requires distinguishing between causes and symptoms. Problems may be:

- **Internal** (a key employee is reassigned to another project)
- **External** (a permit required to begin work is delayed).
- **Technical** (differences of opinion about the best way to design a product)
- **Managerial (**a functional group is not producing according to plan) or
- **Interpersonal** (personality or style clashes).

(e) Decision-making

Decision-making includes analyzing the problem to identify viable solutions, and then making a choice from among them.

Decisions can be made or obtained (from the customer, from the team, or from a functional manager).

Once made, decisions must be implemented.

Decisions also have a time element to them—the "right" decision may not be the "best" decision if it is made too early or too late.

(f) Influencing the Organization

Influencing the organization involves the ability to "get things done."

It requires an understanding of both the formal and informal structures of all the organizations involved—the performing organization, customer, partners, contractors, and numerous others, as appropriate.

Influencing the organization also requires an understanding of the mechanics of power and politics.

Both power and politics are used here in their positive senses.

Power is "the potential ability to influence behavior, to change the course of events, to overcome resistance, and to get people to do things that they would not otherwise do."

In similar fashion, "politics is about getting collective action from a group of people who may have quite different interests. It is about being willing to use conflict and disorder creatively.

The negative sense, of course, derives from the fact that attempts to reconcile these interests result in power struggles and organizational games that can sometimes take on a thoroughly unproductive life of their own."

Influence can be exercised in a: variety of ways. It may be the 'bridge to engine room' approach, where the leader commands and controls. Or the influence can be exercised by guiding and facilitating the group's behavior so that the goal is accomplished.

Finally, leadership implies **that a leader motivates the group to spend energy in attaining the goals of the group**.

Influence without change or movement isn't influence. Leaders make change happen, a difficult but vitally important task.

(g) Mentoring

Mentoring or being a Transformational leadership is more concerned with engagement between leaders and followers.

Leaders attempt to engage the full person of the subordinate and enthuse them. They arouse in their subordinates a heightened awareness of the key issues for the group or the organization. They seek to concern subordinates with achievement, growth and development.

Creating a new vision is something that almost all writers on transformational leadership emphasize.

The vision points the way to a new state of affairs. It is an appealing picture of a more desirable future. It inspires people to believe that the future is worth the upheaval of undoing the present.

A vision needs to be a source of self-esteem and a common purpose for members of the organization.

(h) Technical Skills:

A project manager must have *technical skills*. This relates to financial planning, contract management, and managing creative thinking and problem solving techniques are promoted.

LECTURE # 5

1. Introduction & Fundamentals

1.16 PM's nine Knowledge Areas

1. Project Integration Management

Project Integration Management includes the processes required to ensure that the various elements of the project are properly coordinated. It involves making tradeoffs among competing objectives and alternatives to meet or exceed stake-holder needs.

These processes interact with each other and with the processes in the other knowledge areas as well. Each process may involve effort from one or more individuals or groups of individuals, based on the needs of the project. Each process generally occurs at least once in every project phase.

Project integration management comes into play when a cost estimate is needed for a contingency plan, or when risks associated with various staffing alternatives must be identified. However, for a project to be completed successfully, integration must also occur in a number of other areas as well. For example:

- The work of the project must be integrated with the ongoing operations of the performing organization.
- Product scope and project scope must be integrated.

One of the techniques used to both integrate the various processes and to measure the performance of the project as it moves from initiation through to completion is *Earned Value Management (EVM).*

- *Earned value* is the amount of work completed, measured according to the budgeted effort that the work was supposed to consume.
- It is also called the budgeted cost of work performed.
- As each task is completed, the number of person-months originally planned for that task is added to the earned value of the project.

– **Earned value charts***:* An earned value chart has three curves:

- The budgeted cost of the work scheduled.
- The earned value.
- The actual cost of the work performed so far.

2. Project Scope Management

Project Scope Management includes the processes required to ensure that the project includes all the work required, and only the work required, to complete the project successfully. It is primarily concerned with defining and controlling what is or is not included in the project.

The processes, tools, and techniques used to manage *product* scope vary by application area and are usually defined as part of the project life cycle

A project generally results in a single product, but that product may include subsidiary components, each with its own separate but interdependent product scopes. For example, a new telephone system would generally include four subsidiary components—hardware, software, training, and implementation.

Completion of the project scope is measured against the project plan, but completion of the product scope is measured against the product requirements. Both types of scope management must be well integrated to ensure that the work of the project will result in delivery of the specified product.

3. Project Time Management

Project Time Management includes the processes required to ensure timely completion of the project. The followings are major processes in developing the project time schedule:

(a) **Activity Definition**—identifying the specific activities that must be performed to produce the various project deliverables.

(b) **Activity Sequencing**—identifying and documenting interactivity dependencies.

(c) **Activity Duration Estimating**—estimating the number of work periods that will be needed to complete individual activities.

(d) **Schedule Development**—analyzing activity sequences, activity durations, and resource requirements to create the project schedule.

(e) **Schedule Control**—controlling changes to the project schedule.

These processes interact with each other and with the processes in the other knowledge areas as well. Each process may involve effort from one or more individuals or groups of individuals, based on the needs of the project. Each process generally occurs at least once in every project phase.

4. Project Cost Management

Project Cost Management includes the processes required to ensure that the project is completed within the approved budget.

Resource Planning—determining what resources (people, equipment, materials) and what quantities of each should be used to perform project activities.

Cost Estimating—developing an approximation (estimate) of the costs of the resources needed to complete project activities.

Cost Budgeting—allocating the overall cost estimate to individual work activities.

Cost Control—controlling changes to the project budget.

These processes interact with each other and with the processes in the other knowledge areas as well. Each process may involve effort from one or more individuals or groups of individuals, based on the needs of the project. Each process generally occurs at least once in every project phase.

5. Project Quality Management

Project Quality Management includes the processes required to ensure that the project will satisfy the needs for which it was undertaken. It includes "all activities of the overall management function that determine the quality policy, objectives, and responsibilities and implements them by means such as quality planning, quality assurance, quality control, and quality improvement, within the quality system.

These processes interact with each other and with the processes in the other knowledge areas as well. Each process may involve effort from one or more individuals or groups of individuals, based on the needs of the project. Each process generally occurs at least once in every project phase.

Project quality management must address both the management of the project and the product of the project. The generic term *product* is occasionally used, in literature regarding quality, to refer to both goods and services.

6. Project Human Resource Management

Project Human Resource Management includes the processes required to make the most effective use of the people involved with the project. It includes all the project stakeholders—sponsors, customers, partners, and individual contributors

Following are some major processes:

- **Organizational Planning**—identifying, documenting, and assigning project roles, responsibilities, and reporting relationships.
- **Staff Acquisition—**getting the human resources needed assigned to and working on the project.
- **Team Development**—developing individual and group competencies to enhance project performance.

These processes interact with each other and with the processes in the other knowledge areas as well. Each process may involve effort from one or more individuals or groups of individuals, based on the needs of the project.

There is a substantial body of literature about dealing with people in an operational, ongoing context. Some of the many topics include:

- Leading, communicating, negotiating, etc.

Key General Management Skills:

- Delegating, motivating, coaching, mentoring, and other subjects related to dealing with individuals.
- Team building, dealing with conflict, and other subjects related to dealing with groups.
- Performance appraisal, recruitment, retention, labor relations, health and safety regulations, and other subjects related to administering the human resource function.

Most of this material is directly applicable to leading and managing people on projects, and the project manager and project management team should be familiar with it. However, they must also be sensitive as to how this knowledge is applied on the project. For example:

Project Human Resource Management includes the processes required to make the most effective use of the people involved with the project. It includes all the project stakeholders—sponsors, customers, partners, and individual contributors.

Major processes include:

- **Organizational Planning**—identifying, documenting, and assigning project roles, responsibilities, and reporting relationships.
- **Staff Acquisition**—getting the human resources needed assigned to and working on the project.
- **Team Development**—developing individual and group competencies to enhance project performance.

These processes interact with each other and with the processes in the other knowledge areas as well. Each process may involve effort from one or more individuals or groups of individuals, based on the needs of the project.

7. Project Communications Management

Project Communications Management includes the processes required to ensure timely and appropriate generation, collection, dissemination, storage, and ultimate disposition of project information. It provides the critical links among people, ideas, and information that are necessary for success. Everyone involved in the project must be prepared to send and receive communications, and must understand how the communications in which they are involved as individuals affect the project as a whole.

Major processes include:

- **Communications Planning**—determining the information and communications needs of the stakeholders: who needs what information, when they will need it, and how it will be given to them.
- **Information Distribution**—making needed information available to project stakeholders in a timely manner.
- **Performance Reporting**—collecting and disseminating performance information. This includes status reporting, progress measurement, and forecasting.
- **Administrative Closure**—generating, gathering, and disseminating information to formalize a phase or project completion.

These processes interact with each other and with the processes in the other knowledge areas as well. Each process may involve effort from one or more individuals or groups of individuals, based on the needs of the project. Each process generally occurs at least once in every project phase.

Communicating is a broader subject and involves a substantial body of knowledge that is not unique to the project context. For example:

- **Sender-receiver models**—feedback loops, barriers to communications, etc.
- **Choice of media**—when to communicate in writing versus when to communicate orally, when to write an informal memo versus when to write a formal report, etc.
- **Writing style**—active versus passive voice, sentence structure, word choice, etc.
- **Presentation techniques**—body language, design of visual aids, etc.
- **Meeting management techniques**—preparing an agenda, dealing with conflict, etc.

8. Project Risk management

Project Risk management is the systematic process of identifying, analyzing, and responding to project risk. It includes maximizing the probability and consequences of positive events and minimizing the probability and consequences of adverse events to project objectives.

- **Risk Management Planning**—deciding how to approach and plan the risk management activities for a project.
- **Risk Identification**—determining which risks might affect the project and documenting their characteristics.
- **Qualitative Risk Analysis**—performing a qualitative analysis of risks and conditions to prioritize their effects on project objectives.
- **Quantitative Risk Analysis**—measuring the probability and consequences of risks and estimating their implications for project objectives.
- **Risk Response Planning**—developing procedures and techniques to enhance opportunities and reduce threats to the project's objectives.
- **Risk Monitoring and Control**—monitoring residual risks, identifying new risks, executing risk reduction plans, and evaluating their effectiveness throughout the project life cycle.

These processes interact with each other and with the processes in the other knowledge areas. Each process generally occurs at least once in every project.

9. Project Procurement Management

Project Procurement Management includes the processes required to acquire goods and services, to attain project scope, from outside the performing organization. For simplicity, goods and services, whether one or many, will generally be referred to as a *produc*t.

An overview of the major processes includes:

- **Procurement Planning**—determining what to procure and when.
- **Solicitation Planning**—documenting product requirements and identifying potential sources.
- **Solicitation**—obtaining quotations, bids, offers, or proposals, as appropriate.
- **Source Selection**—choosing from among potential sellers.
- **Contract Administration**—managing the relationship with the seller.
- **Contract Closeout**—completion and settlement of the contract, including resolution of any open items.

These processes interact with each other and with the processes in the other knowledge areas as well. Each process may involve effort from one or more individuals or groups of individuals, based on the needs of the project

*Project Procurement Management is discussed from the perspective of the buyer in the buyer-seller relationshi*p. The buyer-seller relationship can exist at many levels on one project. Depending on the application area, the seller may be called a *subcontracto*r, a *vendo*r, or a *supplie*r.

The *seller* will typically manage its work as a project. In such cases:

- The *buyer* becomes the customer, and is thus a key stakeholder for the seller.
- The seller's project management team must be concerned with all the processes of project management, not just with those of this knowledge area.
- The terms and conditions of the contract become a key input to many of the seller's processes. The contract may actually contain the input (e.g., major deliverables, key milestones, cost objectives), or it may limit the project team's options (e.g., buyer approval of staffing decisions is often required on design projects).

LECTURE # 6

1. Introduction & Fundamentals

1.17 Team leader

Most definitions of leadership involve three components: **influence group and goal**. First, leaders are individuals who *influence* the behavior of others. These others are usually referred to as sub ordinates or followers. Second, leadership is usually examined in the context of a *group,* especially work groups such as managers and their teams or foremen and their subordinates. Third, research on leadership stresses a group *goal* that has to be accomplished. So a definition is:

Leadership is the process in which an individual influences the group members towards the attainment of group or organization goals.

Note that the *influence* can be exercised in a: variety of ways. It may be the 'bridge to engine room' approach, where the leader commands and controls. Or the influence can be exercised by guiding and facilitating the group's behavior so that the goal is accomplished. The notion of *reciprocity* is also part of man definitions. Influencing is often two ways. Leaders may influence followers, but followers influence leaders to lead in one way rather than another. An influencing style appropriate for checkout assistant in a supermarket may be different from that appropriate for rock scientists. The choice is open to the leader of how to influence is one of the key aspects investigated by leadership researchers.

Another aspect of leadership is that the right to lead is often *voluntarily conferred* on the leader by some or all members of the group. A group of friends may recognize one, of their group as the leader, in the sense that she influences the group more than any of the other members. There also may be informal leader. While the nominal head of a department may have the formal leadership position, the real leadership may be exercised by someone lower down the hierarchy who influences the group towards goals that may not be those that the organization whishes to be pursue.

Finally, leadership implies **that a leader motivates the group to spend energy in attaining the goals of the group**. Influence without change or movement isn't influence. Leaders make change happen, a difficult but vitally important task.

1.18 Leaders and Managers

Although it is common to use the terms 'leader' and 'manager' interchangeably, nowadays many writers point to a difference between the two.

The difference is that to function as a leader, a person must exercise influence over another person in the attainment of organizational goals, as described in the definition above. Managerial ' functions of organizing, planning, scheduling, processing information, communicating, and so on, do not necessarily involve leadership. Some managers perform both types of function and can be described as leaders, but others do not. There is no automatic link between the two concepts. Note, too, that leaders are not necessarily just at the top of organizations. Influence can be exerted in most job functions and at levels of seniority or hierarchy. The distinction between managers and leaders was, developed by Bennis and Nanus (1985) in their influential book *Leaders.* In it they put forward the view that:

Leadership is path finding
Management is path following
Management is about doing things right
Leadership is about doing the right things

What they meant by this is that leadership is about having a vision. It involves having a strategy or thinking strategically; it means having a view of where the organization should go or be or do; it means deciding what is important for the success of the organization; it involves envisaging the future, A leader's responsibility is to think what are the key criteria for success of his or her part of the business and not just now but for the future.

Managers, on the other hand, are more concerned with implementing others strategies and plans, They are concerned with running their part of the organization, making sure that the accounts get prepared, that invoices are sent out, that the service is sold, that the traffic is directed, that the research paper is written, or whatever the task that needs to be done.

A very similar view is put forward by Kotter (1990). He argues that management is concerned with activities which are designed to produce 'consistency and order', whereas leadership is concerned with, 'constructive or adaptive change'. Kotter says there are four major ways that management and leadership differ:

1. **Planning and budgeting versus establishing direction.** Management involves making detailed steps and timetables for achieving results, then marshalling resources to make it happen. Leadership means developing a vision *of* the future and strategies for achieving that vision.
2. **Organizing and staffing versus aligning people**. Management comprises the allocation *of* tasks in line with plans, staffing them appropriately, delegating responsibility and monitoring implementation. Leadership involves communicating the vision so that others understand and agree with it.
3. **Controlling and problem-solving versus motivating and inspiring**. Management involves monitoring results of a plan, identifying problems with the plan and then solving them. Leadership involves 'energizing people'

towards the vision. It means appealing to their needs and values so that they overcome barriers to change.

4. **Outcomes: predictability and order, or change.** Management produces predictability and order so that others, such as customers or shareholders can rely on consistent results. Leadership produces change that is often a quantum leap, such as new products or new approaches to managing people, that makes the organization more competitive.

Kirkpatrick and Locke (1991) suggest that the following traits **distinguish leaders from non-leaders:**

- **Drive (achievement. ambition, energy, tenacity, initiative);**
- **Leadership motivation (personalized or socialized); .honesty and integrity;**
- **Self-confidence (including emotional stability);**
- **Cognitive ability (the ability to marshal and interpret a wide variety of information);**
- **Knowledge of the business**

They point out, though, that there is much more to being an effective leader than merely possessing a list of traits. While the traits may provide people with the potential for leadership, it is the capacity to create a vision and implement it that turns the potential into reality.

Strong leadership motivation may sound an obvious trait for a leader. After all, only those who want the weighty responsibilities and grueling pressures of leadership are likely to strive for it. McClelland (1985) distinguishes between two types of power motivation. On the one hand, leaders may be interested in *personalized power,* which describes the motivation of leaders who "seek power for its own sake, who wish to dominate others and are often concerned with the status and trappings of power. The late Robert Maxwell, former owner of the Mirror Group allegedly displayed such traits. On the other hand, leaders who show *socialized power* motivation are more interested in cooperating with others to achieve desired goals. They work with others rather than attempting to dominate or control them. From the point of view of subordinates and the organization as a whole, the leader motivated by socialized power is obviously preferable. On the question of cognitive ability; leaders must be able to gather, integrate and interpret large amounts of information. Many

Many researchers have pointed out that it is not necessary to be brilliant, though; leadership effectiveness is helped by above average intelligence, not genius. Of Kirkpatrick and Locke's six characteristics, some would argue that drive and persistence are much more important than intelligence.

In conclusion, the trait approach has undergone a revival. Recent research suggests that traits do matter. Yet the research shows that there are only a handful

of traits which distinguish leaders from others, and a clear distinction between effective and ineffective leaders has not yet emerged.

In an excellent book of technical leadership, Jerry Weinberg suggests a MOI model of leadership:

Motivation: The ability to encourage (by "push or pull") technical people to produce to their best ability.
Organization: The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.
Ideas or innovation: The ability to encourage people to create and feel creative even when they must work within bounds established for a particular soft- ware product or application.

Weinberg suggests that successful project leaders apply a problem solving management style. That is, a software project manager should concentrate on understanding the problem to be solved, managing the flow of ideas, and at the same time, letting everyone on the team know (by words and, far more important, by actions) that quality counts and that it will not be compromised.

Another view [EDG95] of the characteristics that define an effective project manager emphasizes four key traits:

Problem solving An effective software project manager can diagnose the technical and organizational issues that are most relevant, systematically structure a solution or properly motivate other practitioners to develop the solution, apply lessons learned from past projects to new situations, and remain flexible enough to change direction if initial attempts at problem solution are fruitless.

Managerial identity
A good project manager must take charge of the project. She must have the confidence to assume control when necessary and the assurance to allow good technical people to follow their instincts.

Achievement
To optimize the productivity of a project team, a manager must reward initiative and accomplishment and demonstrate through his own actions that controlled risk taking will not be punished.

Influence and team building
An effective project manager must be able to "read" people; she must be able to understand verbal and nonverbal signals and react to the needs of the people sending these signals. The manager must remain under control in high-stress situations.

1.19 Project Organization

People are managed through an organizational structure. This hierarchical structure is based on the four cornerstones of management: delegation, authority, responsibility and supervision (see Fig.1). Delegation bestows authority, and authority produces (and requires) responsibility. Both authority and responsibility require supervision, and effective supervision requires a suitable organizational structure:

Most projects are organized as teams, with each team assigned specific functions within the project, Different types of project require different types of team structure, as for example a team of junior programmers requires a technical team leader while a team of experts may require only an administrative team leader. It is the project manager's responsibility to select the structure best suited for the project.

Basically an organization is a group of people intentionally organized to accomplish an overall, common goal or set of goals. Business organizations can range in size from two people to tens of thousands.

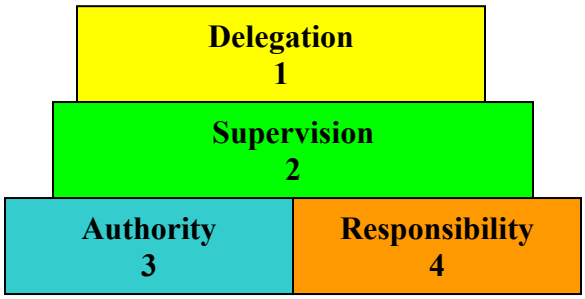


Figure 1: The four cornerstones of management

There are many ways to organize a software project. The larger the project the more critical the organizational structure becomes. Badly organized projects breed confusion, and confusion leads to project failure. Figure 2 describes the basic structure of a project in which below the project manager are just two general functions: development and support. This very basic software project structure was not uncommon in the 1950s and 1960s. It is still a valid project structure for very small projects (up to five developers), though occasionally it can still be found today in larger projects.

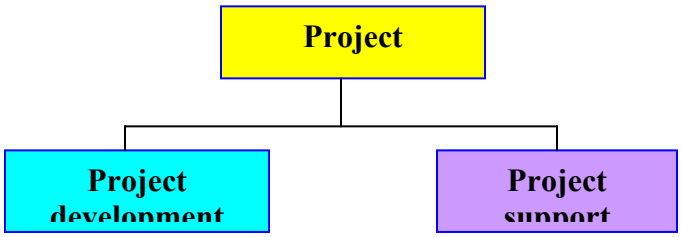


Figure 2: Basic structure of a development project

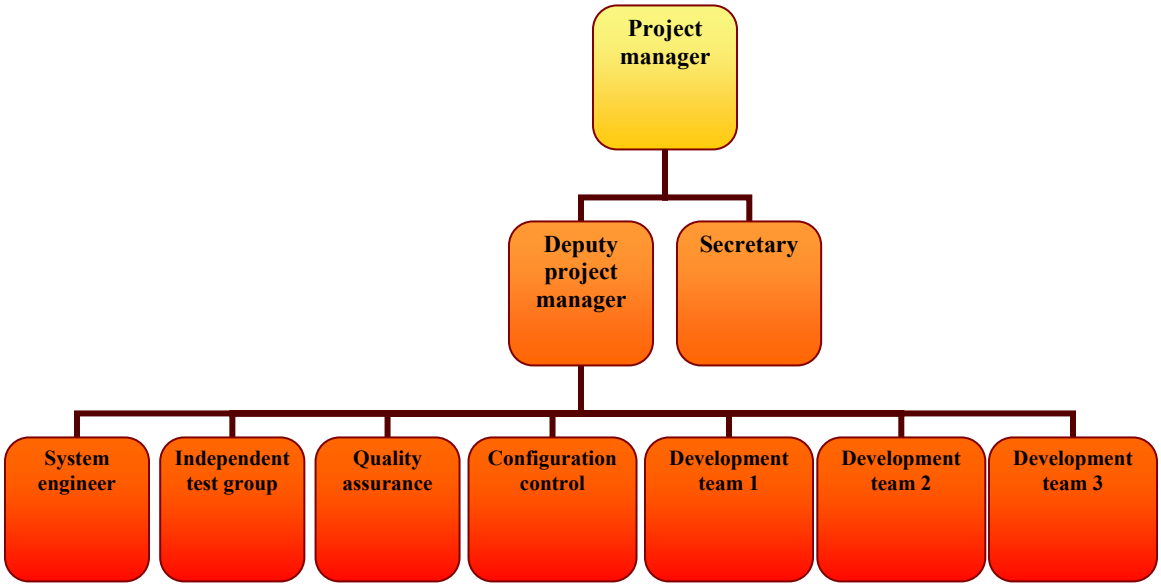


Figure 3: Software project organizational chart

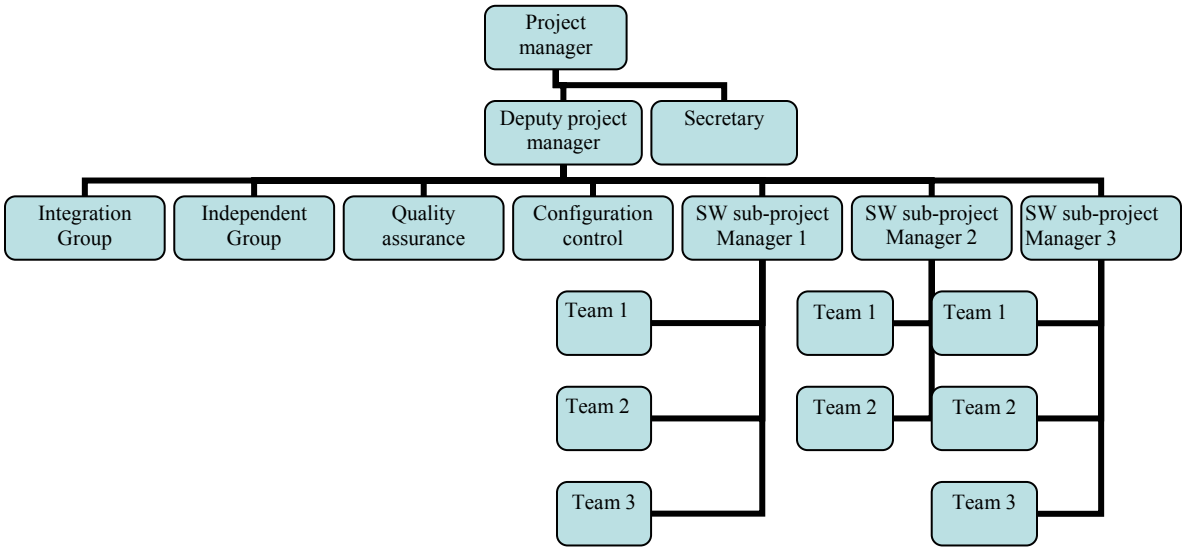


Figure 4: Large hardware/software project organizational chart

Project managers are just two general functions: development and support. This very basic software project structure was not uncommon in the 1950s and 1960s. It is a project structure for very small projects (up to five developers). Though occasionally, it can still be found today in larger projects.

Figure 3 describes a detailed organizational chart including all major support functions. This organizational structure is suitable for large projects (with a staff exceeding 20). Smaller projects may not require a deputy project manager or separate configuration control and quality assurance groups.

Very large projects (exceeding a staff of 40) can often be managed more easily by dividing the project into sub-projects. Figure 4 presents the organizational chart for a large project. This chart includes both software and hardware development teams, and an integration group that is responsible for hardware/software integration as well as integration within each group.

As an example, consider the organization of a large satellite project. The project manager is in fact responsible for a number of projects: the ground control station, the rocket and the satellite itself. The software for all of these sub-projects is managed within a single project office. Each sub-project is then managed by a sub-project manager. An organizational chart similar to the one described in Fig. 4 can be applied to the satellite project; the resulting chart is described in Fig. 5.

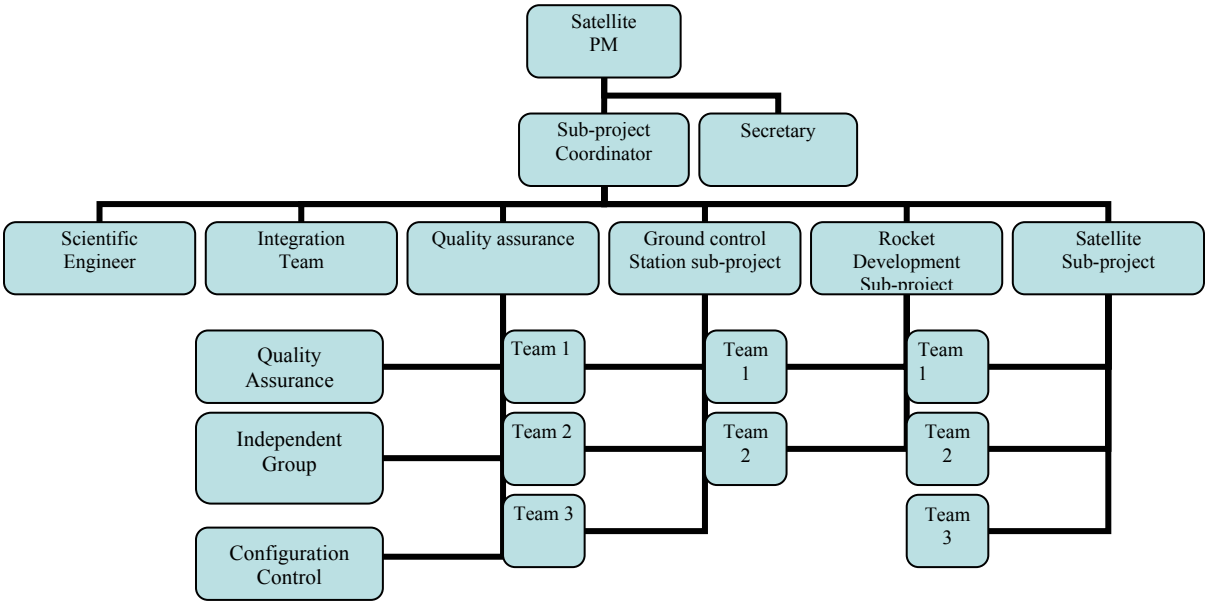


Figure 5: Satellite project organizational chart

Clearly the project's organizational structure is dependent on the type of project being developed. Some of the issues that must be considered are:

- Project size: the larger the project, the more important the organization. Large projects have significant human communications and coordination overhead, and therefore require more support functions.
- Hardware/software development projects. The simultaneous development of hardware and software is not easy. Planning, integration and testing are much more complicated, and require dedicated support groups.
- High reliability systems. Any system that is sensitive to issues of reliability (such as military or life-saving systems) requires a major effort in quality assurance. Quality is also an important consideration in many marketable software products (e.g. communications packages). These types of project require a separate quality assurance organization.

🗁 The Organizational structure

The organization's structure, or design, is the overall arrangement of the organization's various roles, processes and their relationships in the organization. The design of an organization is a *means* to accomplishing the organization's overall goal -- the structure is not an end in itself. In systems theory terms, the design ensures that the appropriate inputs go through the necessary processes to produce the required outputs to produce the intended outcomes.

The structure of the performing organization often constrains the availability of or terms under which resources become available to the project. Organizational structures can be characterized as spanning a spectrum from *functional* to *projectized,* with a variety of matrix structures in between.

⇒ **Corporate structure:** The project's organization is largely dependent on the overall structure of the company within which the project is being developed. Many of the project support functions can be provided by centralized groups within the company. In fact, basic services, such as financial, secretarial and legal services, are commonly provided by the parent or corporate organization.

Corporate structure usually dictates one of two basic types of project organization: matrix or pyramid. Figure 5 describes the structure of a matrix organization (compare this to the pyramid structure in Fig. 4). Within a corporate matrix organization, the project manager manages the technical activities of the project staff, while his or her involvement in non-technical personnel issues (e.g. salary reviews, promotion, and training) is minimal.

⇒ ***Matrix Structure***

Think of the functional structure. Imagine if you took someone from each of the major functions in the functional structure (the boxes along the bottom of the organization chart), e.g., people from sales, engineering, etc., and organized them into a separate group intended to produce and sell one certain kind of product or service. Members of this group stay together until that product is produced or they continue to sell and service it. This overall structure (made up of a functional structure that also has groups assigned to products) is a matrix structure. This structure is useful because it focuses highly skilled people from across the organization to work on a complex product or service. It can be difficult though, because each person essentially reports to two supervisors: the supervisor of the functional area (e.g., engineering) and the product manager, as well. When the organization needs constant coordination of its functional activities, then lateral relations do not provide sufficient integration. Consider the matrix structure. To adopt the matrix structure effectively, the organization should modify many traditional management practices.

Matrix organizations are a blend of functional and projectized characteristics. Weak matrices maintain many of the characteristics of a functional organization, and the

project manager role is more that of a coordinator or expediter than that of a manager. In similar fashion, strong matrices have many of the characteristics of the projectized organization—full-time project managers with considerable authority and full-time project administrative staff.

Most modern organizations involve all these structures at various levels. For example, even a fundamentally functional organization may create a special project team to handle a critical project. Such a team may have many of the characteristics of a project in a projectized organization. The team may include full-time staff from different functional departments, it may develop its own set of operating procedures, and it may operate outside the standard, formalized reporting structure.

Matrix Organizations are a blend of functional and projectized characteristics. Weak matrices maintain many of the characteristics of a functional organization, and the project manager role is more that of a coordinator or expediter than that of a manager. In similar fashion, strong matrices have many of the characteristics of the projectized organization—full-time project managers with considerable authority and full-time project administrative staff.

Most modern organizations involve all these structures at various levels. For example, even a fundamentally functional organization may create a special project team to handle a critical project. Such a team may have many of the characteristics of a project in a projectized organization. The team may include full-time staff from different functional departments, it may develop its own set of operating procedures, and it may operate outside the standard, formalized reporting structure.

- **The advantages of a matrix organization are:**

  - **More expertise**: a matrix organization can maintain experts in specific fields (communications, data bases, graphics etc.) who are then assigned to different projects. A single project cannot always afford the luxury of maintaining experts in all fields.

  - **Flexibility:** it is easier to move people around from one project to another. This results in better utilization of the available expertise.

  - **Emphasis on managing the project**: the project manager is freed of many of the staff management tasks, leaving more time to concentrate on the technical aspects of the project.

  However, matrix organizations also have significant **disadvantages**:

  - **Fewer management measures**. One of the primary tools for generating motivation, promotion, is taken out of the hands of the project manager. The manager has little influence on the developer's salary and professional role in the organization.

- **Lower staff loyalty.** All employees like to know exactly who their superior is. In a matrix organization, an employee has more than one superior. This causes a division of loyalty, and a weaker bond between employee and manager.

These disadvantages often outweigh the advantages of the corporate matrix organization.

Motivation is a major factor in the success of a project, and anything that undermines motivation is usually contrary to the best interests of the project. Unfortunately, the best interests of the project do not always completely coincide with the best interests of the company.

Pyramid organizations provide a clear, well-defined hierarchy in which all individuals know their own position and the positions of those above and below them. When promotion and status play a major role in generating motivation (and they often do), then the pyramid organization is most effective. Many other factors generate motivation; Sense of achievement, praise and peer esteem, are just a few.

Though promotion and status are not always the most effective motivators, a project manager should rarely relinquish any effective management tool. Therefore, from the perspective of a single project, the pyramid organization is often the best.

⇒ ***Functional Structure***

Most business organizations start out with a functional structure, or a small variation of this structure. This is the basic "building block" for other structures. In this structure, there is a central office which oversees various departments or major functions, e.g., human resources, finances, sales, marketing, engineering, etc. Think of a picture that has a box at the top labeled "Central Office". Think of a row of boxes underneath the top box. Each box is labeled, e.g., sales, engineering, human resources, etc. Connect the boxes with lines coming down from the top box to each of the boxes below. Use **functional structures** when the organization is small, geographically centralized, and provides few goods and services. When the organization experiences bottlenecks in decision making and difficulties in coordination, it has outgrown its functional structure.

The classic functional organization is a hierarchy where each employee has one clear superior. Staff members are grouped by specialty, such as production, marketing, engineering, and accounting at the top level, with engineering further subdivided into functional organizations that support the business of the larger organization (e.g., mechanical and electrical). Functional organizations still have projects, but the perceived scope of the project is limited to the boundaries of the function: the engineering department in a functional organization will do its work

independent of the manufacturing or marketing departments. For example, when a new product development is undertaken in a purely functional organization, the design phase is often called a *design project* and includes only engineering department staff. If questions about manufacturing arise, they are passed up the hierarchy to the department head, who consults with the head of the manufacturing department. The engineering department head then passes the answer back down the hierarchy to the engineering project manager.

⇒ ***Project Structure***

In this structure, there is a centralized corporate office and under it, are various divisions each of which is dedicated to producing and / or selling a certain type of business or product, e.g., product 1, product 2, etc. Each division that is dedicated to a certain business or product is, in turn, is organized as its own functional structure. So, for example, the division dedicated to making product 1 has its own sales department, human resources, etc. Basically, project structure is a bunch of functional structures each of which reports to one central office. Use a **divisional structure** when the organization is relatively large, geographically dispersed, and/or produces wide range of goods/services.

In a projectized organization, team members are often collocated. Most of the organization's resources are involved in project work, and project managers have a great deal of independence and authority. Projectized organizations often have organizational units called departments, but these groups either report directly to the project manager or provide support services to the various projects.

🗁 **Structural dimensions:**

- ***Centralization*** -the extent to which functions are dispersed in the organization, either in terms of integration with other functions or geographically
- ***Formalization*** - regarding the extent of policies and procedures in the organization
- ***Hierarchy*** - regarding the extent and configuration of levels in the structure
- ***Routinization*** - regarding the extent that organizational processes are standardized
- ***Specialization*** - regarding the extent to which activities are refined
- ***Training*** - regarding the extent of activities to equip organization members with knowledge and skills to carry out their roles

🗁 **Contextual Dimensions:**

- ***Culture*** - the values and beliefs shared by all (note that culture is often discerned by examining norms or observable behaviors in the workplace)

- ***Environment*** - the nature of external influences and activities in the political, technical, social and economic arenas
- ***Goals*** - unique overall priorities and desired end-states of the organization
- ***Size*** - number of people and resources and their span in the organization
- ***Technology*** - the often unique activities needed to reach organizational goals, including nature of activities, specialization, type of equipment/facilities needed, etc.

🗁 **Role and responsibility assignments**

Project roles (who do what) and responsibilities (who decide what) must be assigned to the appropriate project stakeholders. Roles and responsibilities may vary over time. Most roles and responsibilities will be assigned to stakeholders who are actively involved in the work of the project, such as the project manager, other members of the project management team, and the individual contributors. The roles and responsibilities of the project manager are generally critical on most projects, but vary significantly by application area. Project roles and responsibilities should be closely linked to the project scope definition. A Responsibility Assignment Matrix (RAM) is often used for this purpose. On larger projects, RAMs may be developed at various levels. For example, a high-level RAM may define which group or unit is responsible for each component of the work breakdown structure, while lower-level RAMs are used within the group to assign roles and responsibilities for specific activities to particular individuals.

🗁 **Organization chart**

An organization chart is any graphic display of project reporting relationships. It may be formal or informal, highly detailed or broadly framed, based on the needs of the project. For example, the organization chart for a three- to four-person internal service project is unlikely to have the rigor and detail of the organization chart for a 3,000-person disaster response team. An Organizational Breakdown Structure (OBS) is a specific type of organization chart that shows; which organizational units are responsible for which work packages.

🗁 **Project Personnel**

The staffing management plan describes when and how human resources will be brought onto and taken off of the project team. The staffing plan may be formal or informal, highly detailed or broadly framed, based on the needs of the project. It is a subsidiary element of the overall project plan.

The staffing management plan often includes resource histograms. Particular attention should be paid to how project team members (individuals or groups) will be released when they are no longer needed on the project.

Appropriate reassignment procedures may:

- Reduce costs by reducing or eliminating the tendency to "make work" to fill the time between this assignment and the next.
- Improve morale by reducing or eliminating uncertainty about future employment opportunities.

- **Supporting detail**

Supporting detail for organizational planning varies by application area and project size. Information frequently supplied as supporting detail includes, but is not limited to:

- Job descriptions (position descriptions) —written outlines by job title of the competencies; responsibilities, authority; physical environment, and other characteristics involved in performing a given job.
- Training needs—if the staff to be assigned is not expected to have the competencies

Needed by the project, those competencies will need to be developed as part of the project.

🗁 Key Concepts in Design of Good Organization

Effective Organizational System provides assessment and training in communication effectiveness through focus groups, retreats, workshops, and experiential activities. Coaching addresses such skills as active listening, constructive differing, conflict resolution, negotiation, mediation, persuasiveness, and clarity in delivering a message. At the core of the training is the fundamental premise that the role of communication is to create understanding. When communication is effective, it promotes understanding and plays a fundamental role in building interpersonal skills, leadership and strong teams.

An efficient organization is characterized by timely and productive systems and procedures. At its extreme it has no spare capacity to plan ahead or to respond easily to market changes.

An effective organization is characterized by the ability to pre-empt competitors, respond swiftly and efficiently to changing situations, and display agility in its structure and conduct of operations. It has efficient systems and procedures but has the capacity to anticipate and plan ahead.

Organizational System provides assessment and training in communication effectiveness through focus groups, retreats, workshops, and experiential activities. Coaching addresses such skills as active listening, constructive differing, conflict resolution, negotiation, mediation, persuasiveness, and clarity in delivering a

message. At the core of the training is the fundamental premise that the role of communication is to create understanding. When communication is effective, it promotes understanding and plays a fundamental role in building interpersonal skills, leadership and strong teams. Organizations that achieve high levels of maturity in the people management area have a higher likelihood of implementing effective software engineering practices.

- ***Span of control*** - the range of employees who to report to a managerial position
- ***Authority*** - the formally-granted influence of a position to make decisions, pursue goals and get resources to pursue the goals; authority in a managerial role may exist only to the extent that subordinates agree to grant this authority or follow the orders from that position
- ***Responsibility*** - the duty to carry out an assignment or conduct a certain activity
- ***Delegation*** - process of assigning a task to a subordinate along with the commensurate responsibility and authority to carry out the task
- ***Chain of command*** - the lines of authority in an organization, who reports to whom
- ***Accountability -*** responsibility for the outcome of the process
- ***Line authority -*** the type of authority where managers have formal authority over their subordinates' activities (the subordinates are depicted under the manager on a solid line in the organization chart); departments directly involved in producing services or products are sometimes called line departments
- ***Staff departments -*** the type of authority where managers influence line managers through staff's specialized advice; departments that support or advise line departments are called staff departments and include, e.g., human resources, legal, finance, etc.

LECTURE # 7

2. Software Development Fundamentals Management Fundamentals

2.1 Evolution of Software

The understanding about software and software development has come a long way from the days of punch cards and Ada. In the first stage of computing, hardware mattered the most. Computers themselves were the domains of the government, and most software was developed in defense-funded labs, dedicated to the advancement of science and technology in the national interest.

By the 1950s, large corporations realized the benefits of using computers. They increasingly started using computers to process and analyze financial and production data. These computers were huge in size. There was inadequate software available for them. Even the software that existed was designed essentially to function on a specific hardware product. The software was developed and maintained by the company that manufactured the hardware. Software design and documentation existed only in the developer's head. If the developer left the company, you would find maintenance to be a nightmare.

In the second phase of software evolution, the corporate and academic sectors increasingly started using computers, and the perspective about both software and software development began to change. By the late 1960s and early 1970s, concepts such as multi-sessions, multi-user systems, multiprogramming gained a foothold. Soon computers were developed to collect, process, transform, and analyze data in seconds. The focus of software development shifted from custom software to product software. Now, multiple users on multiple computers could use the same software. This phase of software evolution also saw the emergence of software maintenance activities. These activities comprised fixing bugs, and modifying the software based on changes in user requirements.

The third phase in software evolution was driven by the widespread use of silicon-based microprocessors, which further led to the development of high-speed computers, networked computers, and digital communication. Although, all the advancement in software and hardware was still largely restricted to enterprise applications manufacturers had begun to see the application of the microprocessor in something as mundane as ovens to the robots used in car plants.

The fourth, and current, phase of software evolution began in the early 1990s. This phase saw the growth of client-server environment, parallel computing, distributed computing, network computing, and object-oriented programming.

This phase also witnessed the growing popularity of personal computer (PC). During this phase, the Internet facilitated easy accessibility of information. In addition to complex software to support the advanced hardware, the scope of software development widened to include software products for the common man.

Figure 1.1 sketches the path of software from the 1950s onwards.

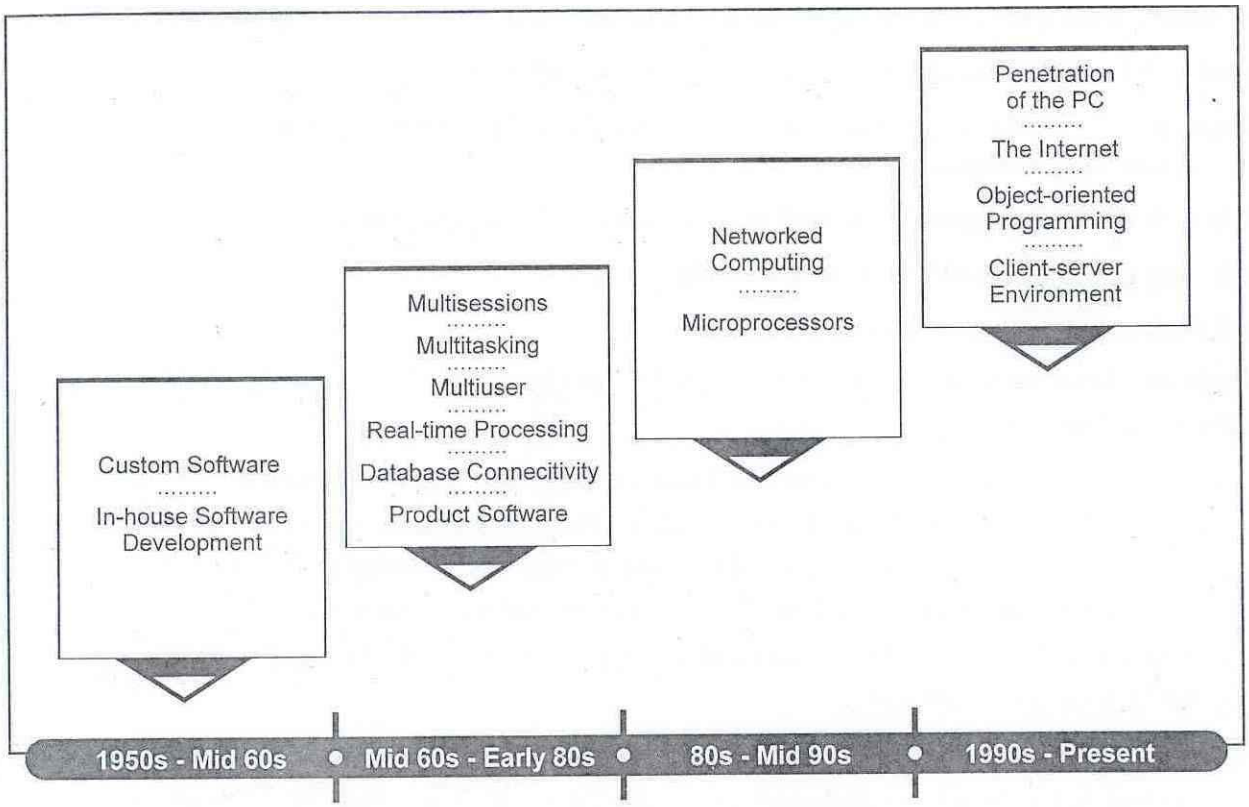


Figure 1.1: Software Evolution

- **The Software Crisis**

The rapid evolution of software design concepts and software development methodology resulted in an ad hoc approach to software development. In the early days of computing, stress was laid more on computer hardware than on software. This happened mainly because hardware consumed the largest portion of the project budget. Project managers closely followed the process of hardware funding, budgeting, analysis and design, production, and implementation. Software development, on the other hand, was left to the developers. There was no training conducted, documentation maintained, or methodology followed for software development. You will find that this attitude persisted until the 1970s, when finally processes and methodologies for software design and development began to be created. However, the ad hoc approach to software development that

was prevalent in the early stages of software evolution hampered the application of systematic processes. This conflict referred to as the software crisis. Some of the reasons to which you can attribute the software crisis include:

- Software developers used-multiple programming languages.
- Software developers used multiple variations of standard programming languages.
- Most of the requirements were complex with regard to the existing capabilities,
- Users, who had little or no experience of developing or even using software, formulated requirements.
- Software developers poorly mapped requirements to the actual product.
- Software developed had low interoperability.
- Software maintenance was costly.
- Hardware developed at a faster rate than software (better hardware requires better software to operate it).

During the period of software crisis, you will find that software that was produced was generally over budgeted, under scheduled, and of poor quality. The immediate knee-jerk response to these problems was software maintenance, which began to consume huge resources. During this period, maintaining software was adopted as a short-term solution due to the costs involved in fixing software regularly. This often resulted in the original software design approach getting lost due to the lack of documentation.

In contrast, the situation in the present times has changed to a large extent. Software costs have risen, although hardware is purchased easily off the shelf. Now, the primary concerns regarding software projects are project delays, high costs, and a large number of errors in the finished product.

2.2 Project Execution Fundamentals

Tracking

For the project manager, it is essential to be constantly informed of the true status of the project. This is achieved by assuring the regular flow of accurate information from the development teams. Many of the methods of acquiring information are not objective and rely on the accuracy of the reports provided by the project developers themselves. They include:

- Periodic written status reports
- Verbal reports
- Status meetings
- Product demonstrations (demos)

Product demonstrations are particularly subjective, because they demonstrate only what the developer wishes to be seen. The project manager needs objective

information. Such information can often be acquired from reports produced by support groups, such as:

- Quality assurance reports
- Independent test reports

Although reports and meetings are indeed useful sources of information, nothing can replace direct contact between the project manager and the development staff. Frequent informal talks with the developers are excellent sources of information, especially when held in an informal atmosphere (and not in the project manager's office).

The project manager must keep on constant guard against an error commonly referred to as the '90/50 syndrome', which states that, 'it takes 50 percent of the time to complete 90 percent of the work, and an additional 50 percent of the time to complete the remaining 10 percent of the work'. This means that project developers will begin to boast quite early that they have 'almost finished' their tasks. Unfortunately, there is a great difference between 'almost finished' and 'finished'.

Finishing a task -writing documentation, and polishing off the last few problems, often takes longer than developers anticipate. This is because these activities produce very few visible results, and developers tend (wrongly) to associate work with results. Therefore, managers can obtain more information from developers by asking them how long they estimate it will take to finish, and not how much of their work has been completed.

- **Status reports**

Status reports should be required from every member of the development team, without exception. The reports should be submitted periodically, usually weekly or bi-weekly, and should contain at least the following three sections (see Fig. 5.8):

1. *Activities during the report period*
   Each subsection within this section describes a major activity during the report period. The description of each activity should span two to three lines. Activities should be linked to the project task list or work breakdown structure (WBS) (see Chapter 10 for a description of the WBS).

2. *Planned activities for the next report period*
   Each subsection within this section describes a major activity planned for the next report period. The description of each activity should span one to two lines.

3. *Problems*

Each subsection within this section describes a major problem that either occurred during the report period, or that was reported previously and has not yet been resolved. This means that problems will be repeatedly reported until they are resolved. In particular, this section must explain why this report's Section 1 does not correspond to the previous report's Section 2.

All reports should also contain:

1. Date of report
2. Report period (e.g. 3 July to 10 July 1992)
3. Name of report (e.g. Communications team status report)
4. Name of person submitting the report

The preparation of a periodic status report should take about 20 minutes, but not longer than 30 minutes. Developers should submit their status reports to their team leader. The team leader then combines the reports of the team into a single status report, while maintaining the same report structure. This activity should take the team leader about 30 minutes, but not longer than 45 minutes (this is easily done when the reports are prepared and submitted by electronic mail).

Each team leader submits the team status report to the project manager. The individual status reports need not be submitted; these should be filed and submitted to the project manager only on request.

From: John Doe, Team leader
To: Frank Smith, Project Manager
Date: 15 June 1993

User interface team: Weekly status report
for the period 5-12 June 1993

1. Activities during the report period:
1.1 The design of the user help screens (activity 3.12.6) was completed on schedule. The design specs were submitted to configuration control.

1.2 Coding of the command pass through modules (activity group 5.12) continues, and is currently behind schedule by about 1 week.

2. Activities planned for next week:
2.1 Coding of the command pass through modules (activity group 5.12) will be completed, and unit tests will be started.

2.2 Two members of the team (Ed and Joan) will attend a two day course on the Programmer's interface to the new user interface package. This is an unscheduled activity that was approved at the last project meeting. This will not delay the schedule, due to the early completion of the command pass through modules (see Section 1.2 above).

3. Problems:
3.1 The user interface package we originally planned to use was found to be inadequate for the project. Two team members will study the new proposed package (sec Section 2.2 above). If the new package is also found to be unsuitable, then this will severely impact our development schedule.

3.2 One of our team members (Jack Brown) has been using an old VTI00 terminal instead of a workstation for the past two weeks, due to the acute shortage of workstations. This is the reason why Jack's task 5.12 was not completed this week, as scheduled.

Figure 5.8: Example of a weekly status report

The project manager also receives status reports from other project support personnel such as the project systems engineer or the deputy project manager. The project manager then prepares the project status report by combining the individual reports received into a single three-part report. The project status report is then submitted to top management.

Project status reports are not necessarily submitted at the same frequency as internal project status reports. Project reports may be submitted bi-weekly or monthly.

- **Project status meetings**

Project status meetings should be held periodically, usually once a week. A good time for status meetings is either at the end of the last day of the week, or at the beginning of the first day of the week. Status meetings also contribute to the atmosphere of order and control within the project, and should be held regularly, at a fixed time. Participants who cannot participate in the project status meeting may, with the project manager's approval, delegate participation to another member of their team.

The project manager prepares for the status meeting by reviewing the status reports submitted by the key project members (particularly scrutinizing the problem section). Therefore the status reports should be submitted at least two to three hours before the status meeting. Project status meetings are attended by the key project members. The meeting begins with a report of project activities and general issues by the project manager. Then each participant should be given about five to ten minutes to report on the activity of his or her team or area of responsibility. The discussion of problems should not be restricted to the person reporting the problem and the project manager. All problems may be addressed by all participants, with possible assistance offered between team leaders, thus making their experience available throughout the project. It is not the project manager's role to provide solutions to the problems, but rather to guide the team members toward solutions.

Solutions should be worked out whenever possible during the status meeting. Any problem not resolved within five minutes should be postponed for discussion by the relevant parties after the status meeting. The proceedings of all project status meetings must be recorded. Verbatim minutes are not required, though the following items should appear in the record:

1. Date of meeting
2. Name of meeting
3. Present (list of participants)
4. Absent (list of absent invited participants)
5. Action items (name, action, and date for completion)
6. Major decisions and items discussed

The record of the project status meeting should be typed and distributed as soon as possible, but no later than by the end of the day. This is particularly important when there are action items to be completed on the same day. When the project is sufficiently large to justify a secretary, then the record will be taken and typed by

the secretary. In smaller projects, the project manager can rotate this task each week between the participants.

2.3 Software Project Management Framework

You all know that a project is much more than a collection of methodologies, tasks, resources, and reviews. A project is a synchronized event where there is perfect harmony and understanding between the participants. The participants are equipped with the essential skills of planning, cooperating, helping, and communicating. However, the most important activity here is to orchestrate the movement of the participants. The onus lies with the project manager to synchronize the activities of the project to result in a perfect presentation. Although each project manager has a unique style of functioning, there are some fundamental approaches that guide a project manager. These approaches are traditional project management concepts and software engineering concepts. To understand software projects and their dynamics, you must be aware of the environment in which a software project is executed. This further requires an understanding of the larger framework of software project management.

In this chapter, you will learn to, build a connection between traditional project management concepts and software engineering concepts. Both traditional and software projects share the same methodologies, techniques, and processes. However, managing software projects requires a distinct approach. In this, chapter, you will learn to apply traditional project management principles to software projects. Further, you will learn about the responsibilities of a software project manager. You will also learn about the phases in a software project and the activities within each phase. Finally, the chapter will provide you an overview of the problems that affect a software project and the myths prevalent about software project management.

2.4 Software Development Life Cycle (SDLC)

The development of a software project consists of many activities spread across multiple phases. Dividing a software project into phases helps you in managing the complexities and uncertainties involved in the software project.

2.5 Phases of a Software project

Each phase represents the development of either a part of the software product or something associated with the software project, such as user manual or testing. Each phase is composed of various activities. You can consider a phase complete when all activities are complete.

A phase is named according to the primary deliverable set that is achieved at the end of that phase. For example, if the requirements document is required as the output, the phase is called the requirements phase. Similarly, most software projects have phases for analysis, design, construction, implementation, and testing.

A typical software project includes the following phases:

- Software requirement analysis phase
- Software Design Phase
- Software Planning Phase
- Software construction phase
- Software testing phase
- Software acceptance and maintenance phase

2.6 SDLC Models

Different organizations have different ways of assessing and arranging the phases in a project. These are called *process models*. Process models define how a software life cycle actually works. They provide you with a framework to plan and execute the various phases in the project. Typically, project life cycles display the following characteristics:

- The level of cost and effort required in a software project life cycle is small to begin with but grows larger towards the end of the project. This happens because the phases such as software construction and implementation, which come at later stages in a software project, require more resources than the initial phases of the project.
- At the start of the SDLC, external entities, such as the customer and the organization, play an important role with regard to their effect on the requirements. However, towards the end of the software project as the cost and effort required to implement changes rise, the requests for change in requirements decreases.
- The uncertainty faced by the software project is highest at the beginning of the SDLC. Their level decreases as the project progresses.

There are a few standard software process models that you can use, with some customization. Some standard process models are given below:

- The *Waterfall* model: This is the traditional life cycle model. It assumes that all phases in a software project are carried out sequentially and that each phase is completed before the next is taken up.
- The *Prototyping* Model: A model that works on an iterative cycle of gathering customer requirements, producing a prototype based on the requirement specifications, and getting the prototype validated by the customer. Each

iteration of the life cycle builds on the prototype produced in the previous iteration.

- The *Incremental* Model: The Incremental model is an example of an evolutionary life cycle model. It combines the linear nature of the Waterfall model and the iterative nature of the Prototyping model. The Incremental model divided the development life cycle into multiple linear sequences, each of which produces an increment of the final software product. In this model, the software product is developed in builds. A build is defined as a self-contained unit of the development activity. The entire development cycle is planned for a specific number of logical builds, each having a specific set of features.
- The *Spiral* model: Another evolutionary life cycle model that combines the linear nature of the Waterfall model and the iterative nature of the Prototyping model. The project life cycle is divided into phases, and each phase is executed in all of the iteration of the Spiral Model.

LECTURE # 8

2. Software Development Fundamentals
Management Fundamentals

2.7 Organizational Issues and Project Management

Organizational issues have a deep influence on a software project, its progress, and the role of the project manager. The policies of an organization can affect the way the organization handles the customer, different types of technologies, and different software projects. The organizational issues that can influence a software project include:

- Reaction to external influences
- Interest in adherence to standards
- Definition of core competency area
- Existence of knowledge management system
- Interest in human resources

One organizational issue that can influence a software project is the reaction of the organization to external influences.- As a project manager, it is important for you to assess how the organization reacts to changes in the external environment, For example, in the current technology environment that changes rapidly, an organization should be proactive in strengthening its capability baseline by adopting new technology and retraining its employees as per market requirements.

Interest in adherence to standards is another organizational issue that can influence a software project. The current technology environment is highly dynamic. Various nonprofit and independent organizations have developed protocols and standards for the standardization of software development and measurement: For example; the Software Engineering Institute at the Carnegie Mellon University has developed the Capability Maturity Model (SEI-CMM). The CMM rates the processes of a software development organization and classifies it into five maturity levels. Software development organizations can also get the quality-related certifications issued by the International Standards Organization smoothen your project management tasks by standardizing the internal processes and optimizing performance.

Definition of core competency area also influences a software project. An organization that creates software must understand how they are created and establish processes accordingly For example, if the core competency of the organization lies in manufacturing chemicals, it should preferably not attempt

software development for such an organization, it is better to purchase an off-the-shelf software product.

An organizational issue that positively influences the tasks of software project managers is the presence of a good knowledge management system within the organization. Knowledge management is the collection of processes that control the creation and utilization of knowledge within the organization. A good knowledge management system allows you to access relevant information and make informed decisions.

Another organizational issue that can influence a software project is the interest of management in human resources. The human resources of an organization are its primary resource. You need to ensure that the people in the development team enjoy a comfortable work environment, which is conducive for smooth and trouble-free work. This includes providing suitable compensations, a friendly work environment, and smooth processes. The absence of these factors negatively affects employee morale, and therefore, productivity.

2.8 Managing Processes

As a software project manager, you become the key player in a software project. You not only manager the day-to-day activities of the project but also ensure that the software product is delivered on time. What makes your role challenging is the performance of project-related activities within a specified budget and time constraint. At the same time, you need to keep the requirements and specifications of the customer in mind.

To deliver expected results, you carry out three successive processes: studying the feasibility of the project, planning to meet the requirement, and executing the plan. These processes can be further broken down into activities, such as planning, staffing, and monitoring.

You plan and organize the software development team to complete development within the stipulated time and budget. To do this, you form a team of people who have the required technical skills. Then, you ensure that all activities are carried out as planned by the relevant people.

Your responsibilities include analyzing customer requirements, determining the scope of the software project, allocating resources top the project, scheduling the project, and executing the project. These responsibilities can be considered in terms of the areas where management skills are required. The primary software project management areas that you need to concentrate on include:

- Managing Resources
- Managing Cost

- Managing Risk
- Managing Schedule
- Managing the project plan
- Managing quality

⇒ **Managing Resources**

The primary input required to create software are resources. Resources for a software project may be of three kinds: human, hardware, "and software. Human resource management is about effectively identifying the people with the appropriate skills, assigning roles and responsibilities to these people, and establishing reporting relationships. On the other hand, hardware and software resource management relates to identifying and ensuring resources such as workstations, disk space on servers, software tools, and software licenses. You need to ensure that human resource identification and allocation is carried out; simultaneously with hardware and software resource management.

To manage resources effectively, there are two areas that you require your attention. These include:

- Management of human resource
- Identification of the critical hardware and software resources

Management of human resources calls for a number of actions. First, you define reporting relationship for the software project. Reporting relationship can exist within and across organizational units, technical areas, and hierarchical levels. Next, determine the skills required for the software project, and identify the appropriate people who possess the required skills. You can review the resource pool and identify resources on the basis of their experience and availability. In case the resources are unavailable, you request for their release from another project or outsource the required resources. Depending upon the organizational practices arid experiences from past projects, you can assign roles and responsibilities to the development team. Finally, create a staff management plan and an organizational chart to show the hierarchical structure of the development team.

Management of human resources also requires efficient team development. This is a complex activity because it combines managing people and organizing the reporting structure within the team. You can build cohesion and commitment within the team through team building activities. These activities include conducting team meetings to involve people from areas other than management into decision-making.

To manage human resources, you also need to implement a reward and recognition system. This helps in promoting and reinforcing positive performance. It is important that you make the link between the performance and

reward explicit and achievable. You should also ensure that the training needs of the development team are met.

A second resource management area that requires attention is identification of the hardware and software resources that are critical for the project. As a software project manager, you must identify all the critical hardware and software resources and document them in the project plan. After the required resources are identified, you define control limits for each resource. Control limits for a resource are the upper and lower limit beyond which the resource is above or below the required level. For example, if a software project requires disk space on the central server, the control limits for the disk space are the required maximum and minimum disk space. Note that not all hardware and software resources are critical to the success of a software project. In some software projects, you may not find any critical resource at the beginning. However, as the project progresses, some resources might become critical. For efficient resource management, you should periodically assess hardware and software resource requirements for the software project. Just as a resource may become critical as a project progresses, a resource might also become less critical over a period of time. Therefore; plan, assess, and take corrective action for all the resources through the duration of the software project.

⇒ **Managing Cost**

The cost factor has a considerable influence on the execution of a software project. The budget of a software project is affected by factors, such as the current orientation of the organization toward software development, number of skilled personnel available, infrastructure, and computer hardware and software. The budget can also be influenced by timely availability of resources. If a particular resource is allocated to the project later than required, the costs involved could go up drastically.

Quite often, when a software project starts to become too expensive, many project managers also tend to start cutting costs. This can have a direct impact on employee morale. When employee morale drops, so does the quality of work, and the productivity. Therefore, as a project manager you must prepare for all circumstances through proper estimation and allocation. To manage costs for software projects, you need an accurate estimation of costs. To do that, there is a sequence of steps that you need to perform.

- Identify the resources required for the project
- Estimate the cost of each activity
- Set cost baselines for each activity
- Implement a control system for cost changes

To estimate costs, you first identify and describe all the resources required in the software project. You also estimate the duration for which the resources are used.

Next, estimate the cost of each resource. To estimate the cost, you can use mathematical tools. However, in the case where limited information is available about resources, you can also use expert judgment to estimate costs.

After the costs of the required resources are estimated, you set cost baselines for each activity. Cost baselines measure the performance of an activity with regard to the cost and duration defined for the completion of the activity.

Finally, as the project manager you implement a control system for cost changes. The cost control system defines cost baselines, identifies cost changes, and modifies cost baselines to adjust cost changes.

⇒ **Management Risk**

Risk management is an integral part of project management. In software projects, where uncertainties are very high, risk management and mitigation is even more critical. Taking risks for high payoffs might bring in high profits but not without the danger of losses. Risk on a small scale is acceptable to most project managers as the element of loss is minimal. However, large risks pose a danger to the progress of a software project and you need to manage them. Risk management activities involve identifying potential risks, assessing them, and planning for contingent actions if a risk materializes.

As a project manager, you perform two primary activities to manage risks for software project:

- Risk Analysis
  - Risk identification
  - Risk quantification
- Risk management

The first activity in risk analysis is risk identification. Risk identification helps you point out the, potential risks for a software project across all phases of the project. Risks might evolve through the duration of a software project, and therefore, risk identification is an ongoing activity. To identify potential risks for a software project, you can analyze the activities in the software project, the software product description, and risks faced by the development team in similar past projects. This exercise allows you to identify the potential sources of risks to the current software project, Assessing the factor influencing the different inputs also allows you to identify the phases in the SDLC when risk might materialize.

After potential risks have been identified, you can quantify them. This is done to ascertain their priority. If multiple risks materialize at the same time, then you must assign a priority to each risk based on the degree of impact on the project and handle the highest-risk events first. For example, the risk of change in client

requirements during the software construction phase is a higher risk than a deadline for a deliverable being missed. Therefore, you first manage the risk that has a higher priority. To quantify project risks, you can use various mathematical and statistical tools. You can also use expert judgment to assess and quantify risks.

After you have identified and quantified the potential risks for a software project, you create a risk mitigation plan. The purpose of the risk mitigation plan is to help you identify procedures to choose the path of least damage and highest returns in a case a risk materializes. To mitigate risks for a software project, you first need to be aware of the opportunities and threats that can be pursued or ignored. This enables you to focus on the risks that might have a negative impact on the software project and develop contingent plans to deal with these risks. You can also mitigate risks by evolving alternative strategies to altogether prevent potential risks from materializing.

In case an unplanned risk materializes, you must be aware of the dependencies of the project activities so that ad hoc solutions can be evolved. However, as the project manager you can avoid unplanned risks from materializing by engaging in an intensive risk identification and mitigation exercise before the software project commences.

⇒ **Managing Schedule**

Time is a major constraint for a software project. With most software projects, the delivery dates for the software product are already committed to the customer at the time the project commences. As a software project manager you must perform various tasks to balance time and deadlines. These are:

- Identify the different deliverables that constitutes the software product
- Define the activities that are required to produce the deliverables
- Identify the interdependencies between activities
- Define the duration of each activity
- Assess the project network diagram
- Create a schedule management and control plan

The first task is to identify the different deliverables that constitute the software product. These deliverables also mark the completion of the different phases within a software project.

Next, define the activities that are required to produce the deliverables. To do this, you can break down the SDLC into phases, identify the deliverables at the end of each phase, and the activities required for creating the deliverables.

After the activities are defined, you identify the interdependencies between them. The purpose of this exercise is to organize the activities and sequence them in the form of a project schedule.

Next, you define the duration of each activity. The inputs that you need for scheduling are the resources required to complete each activity. Then, assesses the availability of these resources and the duration of each similar activity in similar past projects.

After estimating the time required for each activity to complete, you assess the project network diagram. This includes an assessment of the duration estimates, resource requirements, resource pool description, and assumptions and constraints for the software. You can use Mathematical tools to determine a schedule for the project. The project schedule defines the activities within each phase, the team members assigned to complete each activity, the duration of each activity, and the start and end dates for all of the activities.

Finally, create a schedule management and control plan. The purpose of this plan is to identify when changes occur, Implement the changes to the project schedule, and ensure that the changes are beneficial to the software project. After the changes are implemented, you might need to modify the sections of the project plan to ensure that the project is completed on time.

LECTURE # 9

2. Software Development Fundamentals
Management Fundamentals

2.8 Managing Processes

⇒ **Managing the Project Plan**

Preparing it project plan for a software project helps you ensure that the specified requirements and objectives are met successfully. It is a collation of all planning activities that have happened for a software project. This includes activities such as design and analysis, activity definition, risk planning, and cost estimation. To create the plan, you assess all planning activities, organizational policies regarding the creation of the project plan and assumption and constraints for the project. To implement the software project plan, you require management skills, such as leadership, communication, and problem solving, along with the basic knowledge about the software. You also need to ensure that the senior management bf the company has authorized work on the software project. Knowledge management techniques help you to make informed decision regarding the project plan.

After the project plan is executed, you manage the changes to it in such a way that the performance measurement baselines are not impacted, To manage the project plan effectively you monitor the project plan, periodic performance status reports, and requests for change. The primary tool that you can use to control the changes in the project plan is the change control mechanism: This is a set of formal procedures for changing the project plan.

⇒ **Managing Quality**

The quality of software development depends largely on the understanding of ‘quality’ and quality management within the software development team. In software projects where each member has a different understanding of quality and his or her role in implementing it, the software product is usually not of the required quality.

Implementing software quality is often associated with the software developer. However, the role of the project manager is crucial in implementing quality awareness and a quality-producing work environment. When a software project starts missing deadlines and is in danger of exceeding the budget, project managers often lose the focus on quality and lay stress on meeting deadlines. However, the focus should be the other way round. A focus on delivering a

quality product, even at the cost of missing a deadline or two ensures that your customer comes back for repeat orders. On the other hand, the price of non-conformance (PONC) on quality standards simply translates into the loss of goodwill and loss of business.

Modern quality management techniques compliment project management and the role of the project manager. Quality management techniques ensure that a software product conforms to the customer requirements. Quality management further ensures that errors are prevented in the first round itself. Checks for errors and effort for removing the errors consume much more time, effort, and cost than it takes to prevent the errors in the first place. Therefore, maintaining quality in a software project is the domain of all team members.

However, as the project manager you have a critical role in maintaining quality. You provide the resources required to complete the activities of the software project and ensure that quality levels are constantly monitored.

You first identify the areas that must be monitored for ensuring quality and the quality measures to implement. Then, consider the quality policy of the organization, the scope of the project, the applicable standards, and the software product description. Next, use tools, such as a cost-benefit analysis and a flowchart, to identify the areas where you need to monitor quality and the subsequent actions for control. The output of this exercise is the quality plan for the software project. In addition, this exercise allows you to create checklists to help monitor quality.

As a software project manager, you also need to use a quality control mechanism so that the quality of the software product does not suffer. The aim of the quality control mechanism is to ensure compliance with quality standards.

2.9 Project Execution

You already learned in the beginning of the chapter that a software project is divided into different phases. This division is done on the basis of the activities performed in each phase. Similarly, project management activities are also arranged in phases. As a project manager, you perform the activities that map to each of these phases. The project management phases can be broadly categorized as follows;

- Project initiation
- Project closedown
- Project planning, control, and tracking
- Product implementation

- **Project Initiation**

The tasks performed for project initiation are mentioned below:

Requirement gathering: The first task is to gather the customer requirements. Customer requirements may be spoken or unspoken. Therefore, the challenge for the project manager is to elicit the requirements in such a way that both the spoken and unspoken customer needs and wants are gathered. After collecting the required information, you need to translate the customer requirements into technical specifications for the software project.

Scope determination: The scope of a software project can be defined as the combination of the software product arid services to be delivered to the customer. You carry out the scope determination exercise to define the scope of the software project. The scope determination exercise enables you to refine and understand the customer requirements. You can refine the scope definition further by breaking down each deliverable into smaller and more manageable activities. The scope determination exercise also helps you identify the technology for creating the software product.

Resource allocation: During project initiation, you identify the resources required and allocate them to the software project. The resources identified may be people, reusable software components, and hardware or software tools. You allocate the resource to the software project on the basis of the activities defined in the scope determination exercise. While allocating appropriate resources for a software project, you also need to calculate the cost of each resource. The cost of a resource is calculated according to the duration of the resource in the software project. Estimating the cost of resources also helps you prepare a budget for the software project.

Note:
Scope determination and resource allocation are discussed in more detail in later chapters.

Initial project plan: Another exercise that you carry out during project initiation is the creation of a rough project plan. This plan is a draft version and carries only the primitive project plan features. This project plan carries the initial risk analysis of the software project, the initial start and end dates, the duration of the activities in the project, and the sequencing of these activities.

- **Project Planning, Controlling, and Tracking**

This activity of the project manager involves detailed tasks. These tasks are mentioned below:

Detailed project plan: After the scope for the software project is determined and the product design is ready, you prepare a detailed project plan. To create a detailed project plan, you define a detailed list of all the elements that make up the

project deliverables. Next, the deliverables are further broken up to help In the calculation of durations, start dates and end dates for each activity mentioned in the plan. Roles and responsibilities are assigned to people with the appropriate skills to complete each activity within specified time.

Control mechanism: These are set up to control the impact of changes on the software project. The control mechanism includes a detailed risk management and mitigation plan, a detailed quality plan, and quality assurance activities. You also implement a review and audit system for periodic assessment and measurement of the software project activities. The review and audit system enables you to evaluate the progress of the software project. It ensures that all necessary data is collected, deviation from the planned baselines is checked, and corrective action is taken at all checkpoints. In this way, the review and audit system ensures compliance with the organizational processes for software development.

LECTURE # 10

2. Software Development Fundamentals
Management Fundamentals

2.9 Project Execution

⇒ Product Implementation

Product implementation activities involve defining processes related to the implementation of the software product at the customer site. Some of the tasks that you perform for product implementation are mentioned below:

- ***Implementation plan creation***: An implementation plan defines the duration of implementation and the hard and software perquisites for implementing the software product.

- ***Support plan creation***: the project manager also needs to create a support plan for the customer. The support plan includes consideration such as the post-implementation support activities provided to the customer. Post-implementation considerations include the number of support staff available to the customer, their names, contact numbers, and the duration of their availability.

- ***Training plan creation***: During product implementation, you create a training plan to train the customer on the software product. The training plan includes considerations such as the duration of training, the prerequisites for training people, and the number of people that can be trained simultaneously.

- ***User acceptance plan***: You also need to prepare a user acceptance plan. The user acceptance plan provides a detailed outline on how and when the user acceptance tests are performed. The primary focus of user acceptance test is to ensure that the final software product offers all the functionality and performance that the customer wanted. Therefore, the customer tests the software product for issues such as aesthetics, user friendless, and scalability.

⇒ Project Closedown

The final activity for a project manager is project closedown. For most software projects, the project closedown activities take place in the post-implementation phase. However, in some software projects, the customer requests support activities for a longer duration. In such cases, the software project is considered

closed immediately after implementation. The tasks that you perform in project closedown are mentioned below:

- ***Prepare closedown report***: The project closedown report contains the results of the causal analysis that you do for the project. This contains an analysis of what went wrong, what went right, and what you could have done better in the software project.

- ***Identify learning:*** You also need to assess the entire software project and the results of the causal analysis to identify the key learning points from the software project. This helps you identify areas of improvement for future projects. The learning points can also be used by the organization as considerations while planning and executing the next software project.

- ***Identify reusable software components:*** Reusing software components enables you to lower the cost, time, and effort required to complete the software project successfully. After project closedown, you identify the software components that can be reused in future projects of similar nature. The software components prepared for a software project may be complete, partially complete, or in the design stage. These components or their designs can be assessed for usability in future projects.

- ***Create reference material:*** After the project is complete, you can create white papers and reference documents. This can be a significant contribution to the organization and the application area of the software by creating an authoritative knowledge base.

2.10 Project Management Myths

In most cases, you learn the skills required to manage a software project while on the job. As a result, most software project managers practice a lot of management techniques that are of doubtful authenticity. Many software project managers learn about the so-called management skills and concepts that are actually myths.

⇒ Clarifying the Project Management Myths

The following list aims to clarify some of the more prevalent myths in software project management.

1. Combining the best resources with the worst resources available for a software project helps to complete the project successfully.

2. A general statement of objectives is sufficient to begin work on the software project.

3. Allocating extra resources to a late project allows it to catch up with the project schedule.

4. As software by itself is flexible, you can change the requirements at any point in the software project life cycle.

5. The management and the customer always impose an unrealistic deadline for the software project.

6. A software project that meets all the stated objectives is a success.

7. Software maintenance is an easy task and requires less effort than actual software development.

8. Identifying and reporting errors during the reviews makes the software developer unhappy and spoils the work environment.

9. Web-enabling an application or adopting client/server; architecture helps to run software projects smoothly.

Myth: **Combining the best resources with the worst resources available for a project helps to complete the project successfully.**

In software projects, combining the best resources with the worst resources drags down the efficiency and productivity of good resources. This invariably decreases the speed of the software project, and the project ends much after the specified deadline.

Myth: **A general statement of objective is sufficient to begin work on the software project.**

Many software project managers and customers believe that a general statement of objectives gives a reasonable idea of the requirements. However, a formal and detailed description of the customer requirements is needed before the project commences. The software project manager must ensure that all information regarding the software project, such as the functions, performance, interfaces, constraints, assumptions and validation criteria is gathered.

***Myth*:** **Allocating extra resources to a late project allows it to catch up with the project schedule.**

A software project is not a mechanical process such as, say; digging an artificial lake. In case of creating an artificial lake, adding more people to the job can help dig a larger area in the same time. However, in a software project, adding, more people actually increases the time required to finish the project. This happens because a new person joining the project requires time to understand the

requirements of the client, software design, and standards. Moreover, the existing people in the project need to devote time and effort to train the new people on the software project. Therefore, allocating additional resources to a risky situation increases the risk to the software project.

***Myth*: As software by itself is flexible, changes in the requirements can be made at any point in the software project life cycle.**

Requests for changes are common with all projects. However, the timing of the change for requests is critical. This is because an untimely change adversely, impacts the cost of the software project. For example, a change request during the requirements gathering stage has a relatively low impact on costs. On the other hand, a change request during, the software construction stage can be extremely expensive to incorporate. The software project manager must decide with the customer upon a set of objectives that must be achieved at the end of the project. In addition, the project manager and the customer must decide on a specific phase, beyond which only critical change requests are accepted.

***Myth:* The management and customer always impose an unrealistic deadline for the software project.**

The management and customer usually believe that project managers prepare cost effort, and time estimates inclusive of buffers. The management and customers rationalize that if they can cut the buffers by imposing a tight deadline or a low budget on a project, the project manager would still complete the project on time.

***Myth*: A software project that meets all the stated objectives is a success.**

Customer requirements for a software project are always in two forms, spoken and unspoken. Usually, the objectives formed from the customer requirements are based on the spoken requirements. The software project manager must to be aware of the unspoken requirements and ensure that these are met.

***Myth:* Software maintenance is an easy task and requires less effort than actual software development.**

If change requests are made toward the end of the project, then maintenance activities can contribute to large costs and effort overruns. Moreover, contrary to the popular view, implementing changes in the software product in the maintenance stage is a painstaking task.

***Myth:* Identifying and reporting errors during the reviews makes the software developer unhappy and spoil the work environment.**

If a developer makes an error, it is important to point it out so that the error is fixed in time. A project manager must communicate assertively so that the team does not lose focus on quality. In addition, letting an error pass may have ripple effects on the quality of the software product, frustrating the entire team.

Myth: **Web-enabling an application or adopting client / server architecture helps to run software projects smoothly.**

No single technology platform, language, or architecture is a one-point solution for all software projects. All approaches to software development have unique merits and demerits. For example, if a marketing firm needs to make information accessible to people at remote locations, then a, Web-based application is a good option. However, mainframes are still preferred for applications created for the banking industry.

LECTURE # 11

2. Software Development Fundamentals
Management Fundamentals

2.11 Problems in Software Projects

Software projects are similar to traditional projects in the sense that the same types of problems affect them both. However, the difference in managing these problems lies in the approach that you take to the specific issue. For example, a technology-related problem for a software project might be the low degree of reuse of the software components created. However, for a car-manufacturing firm, there is no chance of reusing a component such as a front axle.

You can classify the problems that affect software projects into the following four categories:

- People-related problems
- Process-related problems
- Product-related problems
- Technology-related problems

⇒ **People-related problems**

People-related problems in a software project are:

- ***Low motivation***: As the project manager it is your responsibility to ensure an optimal level of motivation within the team. Lengthy projects, complex activities1 and scarce resources often decrease the motivation level in a software development team. However, you need to lead in such a way that the team is constantly motivated to do a good job.

- ***Problem employees***: Some members of any team always create a problem. For example, an employee may carry a 'holier-than thou' attitude. Problem employees raise the chances of conflicts and differences of opinions within the development team. They lower the efficiency and productivity of other team members and make it difficult to meet the objectives of the software project within the specified time. You need to ensure that employees are not allowed to create a pr9blem for the rest of the team. Even if the employee is very competent, you need to assess the indispensability of such emp1oyees for the project. Moreover, you refrain from playing favorite with certain employees and treat everyone with the same measure.

- ***Unproductive work environment***: The work environment is a major factor that affects the productivity of the development team. For example, a noisy or cramped workspace decreases the motivation levels of the employees. Similarly, unfriendly organizational policies also lower the motivation of the team members. As the project manager, you need to ensure that the team is protected from harmful external make the workspace friendly to work in.

- ***Inefficient project management style:*** the project manager needs to lead by example. The team members *absorb* the work culture, work ethic, and attitude of the project manager and implement it in their work style. If you display a lack of leadership qualities and weak ideals, the motivation levels decrease across software team.

- ***Lack of stakeholder interest:*** For a software project to be a success, each stakeholder needs to take an active interest in the progress of the project. Al1 stakeholders, including the customer, the management, and the software development team, need to commit to the success of the project. For example, if the software development team is not committed to the project, then their contribution may not be to the optimum level.

- ***Ineffective project sponsorship by management:*** Lack of commitment of the senior management to a software project lowers the motivation level of the team members. If the management commits to the progress of a software project, and takes a keen interest in the progress, the confidence of the software development team will increase.

⇒ **Process- related Problems**

The process-related problems in a software project are:

- ***Unrealistic schedule***: Assigning unrealistic deadlines for a software project is a primary reason why software projects are delayed. Often, the marketing or the management team commit a delivery date to the customer in the hope of getting the project contract. However, these dates are not decided in consultation with the development team. The rationale for assigning the deadlines is unfounded. You need to ensure that the deadlines match the ability of the software team to deliver the software product. As it is not always possible to shift deadlines committed to the customer, you also need to plan the resource allocation and project execution such that the deadlines are met.

- ***Insufficient identification:*** Unidentified, partially identified, and unplanned risks pose a threat to the success of a software project. You need to intensively identify risks and evolve a risk management plan such that the project is completed successfully, on time.

- ***Unsuitable life cycle model selection***: Different software projects require different SDLC models. For example, a project to create banking .software is different from software for a satellite where the concept needs to be researched. For the former example, the Waterfall model is more applicable. For the latter example, the Spiral model is more suitable. Selecting the correct life cycle model is critical to the success of a software project.

- ***Abandoning quality under pressure of deadlines:*** Where a software project faces a shortage of resources, time, and funds, project managers often push away quality concerns and focus on meeting deadlines and staying within the budget. Abandoning quality has a ripple effect that actually adds even more time, effort, and costs to the software projects. The cost of doing things right the first time is lower than the cost of inspection during product delivery. Also, the cost of inspection is lower than the cost of debugging software after the customer spots errors.

- ***Unstructured and hurried software development:*** When software project progresses with more focus on meeting deadlines and staying within a budget, the approach to the software development is unstructured and hurried. You should plan the software project such that all the activities are identified, sequenced properly, and roles and responsibilities assigned to the various people on the project. You should also maintain the focus of the development team toward a structured approach to software development.

⇒ **Product-related Problems**

There are many product-related problems that you can face in a software project. These are:

- ***Product scope changed toward the end of the project life cycle***: The project time, effort, and cost estimates for a software project can go up dramatically when the customer changes the scope 9f the product toward the end of the project. In such situations, you should verify the criticality of the scope change. However, if the change request is not critical, you should retain the original scope with a proper explanation to the customer. If the change request is critical, you should explain the situation to the customer. Usually, a customer gives more time and funds to a software project if proper justification is provided. In some cases, the scope change may also be because of a change in government policy. It may become mandatory for you to include such change requests.

- ***Research-oriented software development:*** Many software projects digress from the original scope because of the nature of the software product or technology used. When a totally new kind of software is developed or a new

technology is used, the software development team can lose focus of the objectives by getting into a research-oriented approach. It becomes your responsibility as the project manager to maintain the focus on the objective.

LECTURE # 12

2. Software Development Fundamentals
Management Fundamentals

2.10 Problems in Software Projects

⇒ **Product-related Problems**

- ***III-defined scope:*** You need to define the scope of the software product in the initial stages of a software project. The scope of a software product is defined in terms of the functionality requirements, the performance requirements, the assumptions, and the constraints on the product. If the product scope is ill defined, the software project does not have a proper focus on the features required in the product.

- ***Fuzzy users:*** You also need to clarify the background characteristics of the users of the final software product at the beginning of the software project. If the description of the users is fuzzy, then the software analysis, design, and development stages may reflect the ambiguity with regard to the functions and performance of the final software product.

⇒ **Technology-related problems**

You may also encounter technology-related problems in a software project. These include:

- ***Overestimated savings from reusable components and new tools and methods:*** You can reuse software components in a software project to save time, effort, and cost of creating the component again. It is important that you assess the savings that the use of such a software component provides to a software project. This expectation of both the customer and the management might not be met, if you overestimate the savings from reusing software components.

- ***Switching tools in mid way:*** The current technology environment offers new tools and technologies for software development at a fast rate. All these tools and technologies offer the benefits of a shorter development cycle, lower costs, and under better functionality than earlier tools. You should identify and commit to the tool and technology for the software project before the project commences. Switching the tool or technology used during the software development stage causes the developers to relearn a new tool. In addition,

there is a chance that it might not be possible to integrate the software already developed with the new tool.

- ***Integrating different software products in cross-platform implementation:*** The modem software environment requires that all software should integrate with each other. However, many software projects do not plan for integration with existing software in the same or different domain. This limits the applicationofsu9h software and reduces the shelf life drastically. They key to the success of a software product is interoperability. The software project manager needs to determine the scope for the software product such that is can be integrated easily with existing software.

⇒ **Summary**

The phases in a software project can be organized into a project life cycle. Some standard life cycle models are the Waterfall model, the Prototyping model, the Incremental model, and the Spiral model.

Organizational policies and attitudes influence the progress of a software project and the tasks of a software project manager. For smooth progress of software, project, the organization should be proactive in adopting changes in technology and market environments, focused on developing software, and accept software projects that match the organizational capability baseline. In addition, the organization should implement employee-friendly human resource policies and good knowledge management system.

The role of a software project manager includes managing resources, cost, risk, schedules, project plan, and quality. Software project management activities can be divided into phases. The main phases and the associated activities are initiating the project, planning, controlling, and tracking, implementing the product, and project closedown.

Initiating the project includes gathering requirements, determining the scope, allocating resources, and creating an initial project plan. Planning, controlling, and tracking involve creating a detailed project plan, constructing software, and implementing a control mechanism. Implementing the product comprises implementation plan, support plan, training plan, and user acceptance plan. Project closedown includes preparing closedown report, identifying learning for future projects, and identifying reusable software components for future software projects.

Problems- affecting software projects can be classified into people-related, project-related, product-related, and technology-related.

Various myths regarding software project management are adding more people to a late project can help to finish the project on time, combining the best resources

with the worst resources results in optimal resource allocation, and changes to the scope of the software project and the software product can be made at any-time in the SDLC.

LECTURE # 13

2. Software Development Fundamentals
Technical Fundamentals

2.11 Requirements Management

⇒ **Preview**

Software requirements engineering is a process of discovery, refinement, modeling and specification. The system requirements and role allocated to software-initially established by the system engineer-are refined in detail. Models of the required data information and control flow and operational behavior are created. Alternative solutions are analyzed and a complete analysis model is created.

Requirements engineering is the systematic use of proven principles, techniques, languages, and tools for the cost effective analysis, documentation, and on-going evolution of user needs and the specification of the external behavior of a system to satisfy those user needs. Notice that like all engineering disciplines, requirements engineering is not conducted in a sporadic random or otherwise haphazard fashion, but instead is the systematic use of proven approaches.

Both the software engineer and customer take an active role in software requirements engineering-a set of activities that is often referred to as *analysis*. The customer attempts to reformulate a sometimes nebulous system-level description of data, function and behavior into concrete detail. The developer acts as interrogator, consultant, problem solver and negotiator.

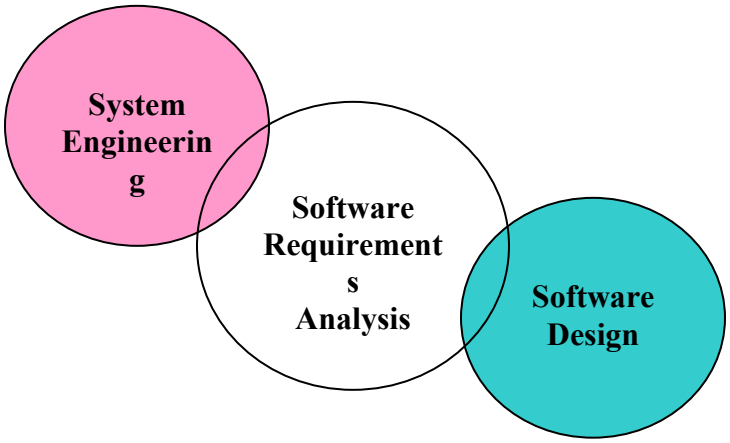
The overall role of Software in a larger system is identified during system engineering. However, it's necessary to take a harder look at software's role-to understand the specific requirements that must be achieved to build high-quality software. That's the job of software requirements analysis. To perform the job properly, you should follow a set of underlying concepts and principles. Generally, a software engineer performs requirements analysis However, for complex business applications a 'system analyst' trained in the business aspects of the application domain may perform the task. If you don't analyze, it's highly likely that you'll build a very elegant software solution that solves the wrong problem. The result is wasted time and money, personal frustration and unhappy customers.

Data, functional, and behavioral requirements are identified by eliciting information from the customer. Requirements are refined and analyzed to assess their clarity, completeness, and consistency.

⇒ **Requirements analysis**

Requirements analysis is a software engineering task that bridges the gap between system level requirements engineering and software design (Figure 1). Requirements engineering activities result in the specification of software's operational characteristics (function, data; and behavior), indicate software's interface with other system elements, and establish constraints that software must meet. Requirements analysis allows the software engineer (sometimes called analyst in this. role) to refine the software allocation and build models of the data, functional, and behavioral domains that will be treated by software. Requirements analysis provides the software designer with a representation of information, function, and behavior that can be translated to data, architectural, interface, and component-level designs. Finally, the requirements specification provides the developer and the customer with the means to assess quality once software is built.

Figure 1: a bridge between system engineering and software design



Software requirements analysis may be divided into five areas of effort:

(1) Problem recognition,
(2) Evaluation and synthesis,
(3) Modeling
(4) Specification, and
(5) Review

Initially, the analyst studies the System Specification (if one exists) and the Software Project Plan. It is important to understand software in a system context and to review the software scope that was used to generate planning estimates. Next, communication for analysis must be established so that problem recognition is ensured. The goal is recognition of the basic problem elements as perceived by the customer/users.

- **Problem evaluation**

Problem evaluation and solution synthesis is the next major area of effort for analysis. The analyst must define all externally observable data objects, evaluate the flow and content of information, define and elaborate all software functions, understand software behavior in the context of events that affect the system, establish system interface characteristics, and uncover additional design constraints. Each of these tasks serves to describe the problem so that an overall approach or solution may be synthesized. For example, an inventory control system is required for a major supplier of auto parts. The analyst finds that problems with the current manual system include:

(1) Inability to obtain the status of a component rapidly
(2) Two or three-day turnaround to update a card file
(3) Multiple reorders to the same vendor because there is no way to associate vendors with components, and so forth.

Once problems have been identified, the analyst determines what information is to be produced by the new system and what data will be provided to the system. For instance, is the customer desires a daily report that indicates what parts have been taken from inventory and how many similar parts remain. The customer indicates that inventory clerks will log the identification number of each part as it leaves the inventory area.

- **Solution synthesis**

Upon evaluating current problems and desired information (input and output), the analyst begins to synthesize one or more solutions. To begin, the data objects processing functions and behavior of the system are defined in detail. Once this information has been established, basic architectures for implementation are considered.

A client/server approach would seem to be appropriate, but does the software to support this architecture fall within the scope outlined in the Software Plan? A database management system would seem to be required, but is user/customer's need for associativity justified? The process of evaluation and synthesis continues until both analyst and

customer feel confident that software can be adequately specified for subsequent development steps.

Throughout evaluation and solution synthesis, the analyst's primary focus is on "what, not "how." What data does the system produce and consume what functions the system must perform. What behaviors do the system exhibit, what interfaces, are defined and what constraints apply?

During the evaluation and solution synthesis activity, the analyst creates models of the system in an effort to better understand data and control flow, functional processing, operational behavior, and information content. The model serves as a foundation for software design and as the basis for the creation of specifications for the Software. The customer may be unsure of precisely what is required. The developer may be unsure that a specific approach will properly accomplish function and performance. For these, and many other reasons, an alternative approach to requirements analysis, called Prototyping, may be conducted.

LECTURE # 14

2. Software Development Fundamentals
Technical Fundamentals

2.11 Requirements Management

⇒ **Requirements Analysis**

- **Evaluation and Synthesis**

  Upon evaluating current problems and desired information (input and output), the analyst begins to synthesize one or more solutions. To begin, the data objects processing functions and behavior of the system are defined in detail. Once this information has been established, basic architectures for implementation are considered.

  A client/server approach would seem to be appropriate, but does the software to support this architecture fall within the scope outlined in the Software Plan? A database management system would seem to be required, but is user/customer's need for associativity justified? The process of evaluation and synthesis continues until both analyst and customer feel confident that software can be adequately specified for subsequent development steps.

  Throughout evaluation and solution synthesis, the analyst's primary focus is on "what, not "how." What data does the system produce and consume what functions the system must perform. What behaviors do the system exhibit, what interfaces, are defined and what constraints apply?

- **Models**

  During the evaluation and solution synthesis activity, the analyst creates models of the system in an effort to better understand data and control flow, functional processing, operational behavior, and information content. The model serves as a foundation for software design and as the basis for the creation of specifications for the Software.

- **Specification**

During the evaluation and solution synthesis activity, the analyst creates models of the system in an effort to better understand data and control flow, functional processing, operational behavior, and information content. The model serves as a foundation for software design and as the basis for the creation of specifications for the Software. The customer may be unsure of precisely what is required. The developer may be unsure that a specific approach will properly accomplish function and performance. For these, and many other reasons, an alternative approach to requirements analysis, called Prototyping, may be conducted.

- **Concerns for Review**

The customer may be unsure of precisely what is required. The developer may be unsure that a specific approach will properly accomplish function and performance. For these, and many other reasons, an alternative approach to requirements analysis, called Prototyping, may be conducted.

For example, an inventory control system is required for a major supplier of auto parts. The analyst finds that problems with the current manual system include:

(1) Inability to obtain the status of a component rapidly,
(2) Two- or three-day turn- around to update a card file,
(3) Multiple reorders to the same vendor because there is no way to associate vendors with components, and so forth.

Once problems have been identified, the analyst determines what information is to be produced by the new system and what data will be provided to the system. For instance, customer desires a daily report that indicates what parts have been taken from inventory and how many similar parts remain. The customer indicates that inventory clerks will log the identification number of each part as it leaves the inventory area.

Upon evaluating current problems and desired information (input and output), the analyst begins to synthesize one or more solutions. To begin, the data objects, processing functions, and behavior of the system are defined in detail. Once this information has been established, basic architectures for implementation are considered.

A client/server approach would seem to be appropriate, but does the software to support this architecture fall within the scope outlined in the *Software Plan?* A database management system would seem to be required, but is the user/customer's need for

associativity justified? The process of evaluation and synthesis continues until both analyst and customer feel confident that software can be adequately specified for subsequent development steps.

⇒ **Requirements Elicitation for Software**

Before requirements can be analyzed, modeled, or specified they must be gathered through an elicitation process. A customer has a problem that may be amenable to a computer-based solution. A developer responds to the customer's request for help.

Communication has begun. But, as we have already noted, the road from communication to understanding is often full of potholes.

1. **Initiating the Process**

The most commonly used requirements elicitation technique is to conduct a meeting or interview. The first meeting between a software engineer (the analyst) and the customer can be likened to the awkwardness of a first date between two adolescents. Neither person knows what to say or ask; both are worried that what they do say will be misinterpreted; both are thinking about where it might lead (both likely have radically different expectations here); both want to get the thing over with, but at the same time, both want it to be a success. Yet, communication must be initiated. Gause and Weinberg [GAU89] suggest that the analyst start by asking context-free questions. That is, a set of questions that will lead to a basic understanding of the problem, the people who want a solution, the nature of the solution that is desired, and the effectiveness of the first encounter itself. The first set of context-free, questions focuses on the customer, the overall goals, and the benefits. For example; the analyst might ask:

- Who is behind the request for this work?
- Who will use the solution?
- What will be the economic benefit of a successful solution?
- Is there another source for the solution that you need?

These questions help to identify all stakeholders who will have interest in the software to be built. In addition, the questions identify the measurable benefit of a successful implementation and possible alternatives to custom software development.

The next set of questions enables the analyst to gain a better understanding of the problem and the customer to voice his or her perceptions about a solution:

- How would you characterize "good" output that would be generated by a successful solution?
- What problem(s) will this solution address?
- Can you show me (or describe) the environment in which the solution will be used?
- Will special performance issues or constraints affect the way the solution is approached?

The final set of questions focuses on the effectiveness of the meeting. Gause and Weinberg call these meta-questions and propose the following (abbreviated) list:

- Are you the right person to answer these questions? Are your answers "official"?
- Are my questions relevant to the problem that you have?
- Am I asking too many questions?
- Can anyone else provide additional information?
- Should I be asking you anything else?

These questions (and others) will help to "break the ice" and initiate the communication that is essential to successful analysis. But a question and answer meeting format is not an approach that has been overwhelmingly successful. In fact, the Q&A session should be used for the first encounter only and then replaced by a meeting format that combines elements of problem solving, negotiation, and specification.

2. Facilitated Application Specification Techniques

Too often, customers and software engineers have an unconscious "us and them" mind-set. Rather than working as a team to identify and refine requirements, each constituency defines its own "territory" and communicates through a series of memos, formal position papers, documents, and question and answer sessions. History has shown that this approach doesn't work very well. Misunderstandings abound, important information is omitted, and a successful working relationship is never established.

It is with these problems in mind that a number of independent investigators have developed a team-oriented approach to requirements gathering that is applied during early stages of

analysis and specification. Called facilitated application specification techniques "(FAST).

This approach encourages the creation of a joint team of customers and developers who work together to:
Identify the problem
Propose elements of the solution
Negotiate different approaches and specify a preliminary set of solution requirements.

FAST has been used predominantly by the information systems community, but the technique offers potential for improved communication in applications of all kinds.

Many different approaches to FAST have been proposed. Each makes use of a slightly different scenario, but all apply some variation on the following basic guidelines:

- A meeting is conducted at a neutral site and attended by both software engineers and customers.
- Rules for preparation and participation are established.
- An agenda is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas.
- A ‘facilitator’ (can be a; customer, a developer, or an outsider) controls the meeting.
- A "definition mechanism" (can be work sheets, flip charts, or wall stickers or an electronic bulletin board, chat room or virtual forum) is used.

The goal is to identify the problem, propose elements of the solution, negotiate different approaches, and specify a preliminary set of solution requirements in an atmosphere that is conducive to the accomplishment of the goal.

To better understand the flow of events as they occur in a typical FAST meeting, we present a brief scenario that outlines the sequence of events that lead up to the meeting, occur during the meeting, and follow the meeting.

Initial meetings between the developer and customer occur and basic questions and answers help to establish the scope of the problem and the overall perception of a solution. Out of these initial meetings, the developer and customer write a one- or two-page "product request." A meeting place, time, and date for FAST are selected and a facilitator is chosen. Attendees from both the

development and customer/user organizations are invited to attend. The product request is distributed to all attendees before the meeting date.

3. Quality Function Deployment

A quality management technique that translates needs of customers into technical requirements of software.

i. Normal Requirement: meeting objectives & goals stated for a product or system during meeting

i. Expected Requirement: Implicit to products / system and may be so fundamental that customer does not explicitly state them

i. Exciting Requirement: Features beyond customer's expectation and prove to be very satisfying when present

4. Use Cases

As requirements are gathered as part of:

- Informal meeting
- FAST or QFD

SW Engineer can create a set of scenario that identify a thread of usage for system to be constructed; providing a description of how system will be used.

5. Analysis Principles

A variety of modeling notations are developed by investigators. Each analysis method has a unique point of view. However all analysis methods are related by a set of operational principles like:

– The information domain of a problem must be represented and understood.

– The functions that the software is to perform must be defined.

– The behavior of the software (as a sequence of external events) must be represented.

– The models that depict information function and behavior must be partitioned in a manner that uncovers details in a layered (or hierarchical) fashion.

- The analysis process should move from essential information toward implementation detail.

6. Software Prototyping

Analysis should be conducted regardless of the SW engineering paradigm. (Various approaches apply)

In some cases it is possible to apply operational analysis principles and derive a model of SW from which a design can be developed.

In other situation Requirement Elicitation (FAST, QFD etc) is conducted and a model is built, called Prototype.

LECTURE # 15

2. Software Development Fundamentals
Technical Fundamentals

2.11 Requirements Management

⇒ The Software Requirements Specification

The *Software Requirements Specification* is produced at the culmination of the analysis task, The function and performance allocated to software as part of system engineering are refined by establishing a complete information description, a detailed functional description, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria, and other: information pertinent to requirements. The National Bureau of Standards; IEEE (Standard No. 830-1984), and the U.S. Department of Defense have all proposed candidate formats for software requirements specifications (as well as other software engineering documentation).

Mode of specification has a great impact on quality of solution. Forcing SWE to work with incomplete, inconsistence, or misleading specifications result in frustration and confusion affecting:

- ***Quality***
- ***Timeliness and***
- ***Completeness of SW product***

- **Principles**

i. Separate functionality from implementation.

ii. Develop a model of desired behavior of a system that encompasses data and the functional response of a system to various stimuli from the environment.

iii. Establish the context in which SW operates by specifying the manner in which other system components interact with software.

iv. Define the environment in which system operates and indicate how a highly inter-wined collection of agents react to stimuli in the environment (changes to objects) produced by those agents.

v. Create a cognitive model rather than a design or implementation model. Cognitive model describes a system as perceived by its user community.

vi. Recognize that; "the specifications must be tolerant of incompleteness and augmentable."

vii. A specification is always a model –an abstraction-of some real (or envisioned) situation that is normally quite complex. Hence it will be incomplete and will exit at many levels of detail.

i. Establish the content and structure of a specification in a way that will enable it to be amenable to change.

- **Representation**

  i. Representation format and content should be relevant to the problem. (A general outline of SRS can be developed).

  ii. Information contained within the specification should be nested.

  iii. It should reveal layers of information so that reader can move to the level of detail required. Paragraph & diagram number scheme to indicate level.

  iv. Diagrams and other notational forms should be restricted in number and consistent in use. (Confusing or inconsistent notations, whether graphical or symbolic degrades understating and foster errors.)

  v. Representation should be revisable.

  vi. The content of specification will change. Ideally, CASE tools should be available to update all representations that are affected by each change.

  The function and performance allocated to software as part of system engineering are refined by establishing:

  1. A complete information description,
  2. A detailed functional description,
  3. A representation of system behavior,
  4. An indication of performance requirements and design constraints,
  5. Appropriate validation criteria, and other: information pertinent to requirements.

  The *Introduction* of the software requirements specification states the goals and objectives of the software, describing it in the context of the computer-based system; actually, the Introduction may be nothing more than the software scope of the planning document.

The *Software Requirements Specification* is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing:

- A complete information description
- A detailed functional description
- A representation of system behavior
- An indication of performance requirements and design constraints
- Appropriate validation criteria and other information pertinent to requirements.

The *Information Description* provides a detailed description of the problem that the software must solve. Information content, flow, and structure are documented. Hardware, software, and human interfaces are described (or external system elements: and internal software functions.

A description of each function required to solve the problem is presented in the *Functional Description.*

A processing narrative is provided for each function:

- *Design constraints are stated and justified*

- *Performance characteristics are stated, and*

- *One or more diagrams are included to graphically represent the overall structure of the software and interplay among software functions and other system elements*

The *Behavioral Description* section of the specification examines the operation of the software as a consequence of external events and internally generated control characteristics.

Validation Criteria is probably the most important and, ironically, the most often, neglected section of the *Software Requirements specification.*

- How do we recognize a successful implementation?
- What *classes* of tests must be conducted to validate function, performance, and constraints?

We neglect this section because completing it demands a thorough understanding of software requirements-something that we often do not have at this stage.

Yet, specification of validation criteria acts as an implicit review of all other requirements. It is essential that time and attention be given to this section.

Finally, the specification includes a *Bibliography and Appendix.*

- ***The bibliography*** contains references to all documents that relate to the software. These include other software engineering documentation, technical references, vendor literature, and; standards.
- ***The appendix*** contains information that supplements the specifications. Tabular data, detailed description of algorithms, charts, graphs and other material, are presented as appendixes.

In many cases the *Software Requirements Specification* may be accompanied by an executable prototype (which in some cases may replace the specification), a paper prototype or a *Preliminary User's Manual.* The *Preliminary Users Manual* presents the software as a black box: That is, heavy emphasis is placed on user input and the resultant output. The manual can serve as a valuable tool for uncovering problems at the human/machine interface.

⇒ **Review**

A review of the *Software Requirements Specification* (and/or prototype) is conducted by both the software developer and the customer.

Extreme care should be taken in conducting the review because the specification forms the foundation of the development phase.

The review is first conducted at a macroscopic level; that is, reviewers attempt to ensure that the specification is complete, consistent, and accurate when the overall information, functional, and behavioral domains; are considered. However, to fully explore each of these domains, the review becomes more detailed, examining not only broad descriptions but the way in which requirements are worded. For example, when specifications contain "vague terms" (e.g., *some, sometimes, often, usual ordinarily, most,* or *mostly),* the reviewer should flag the statements for further clarification.

Once the review is complete, the Software Requirements Specification is "signed-off by both the customer and the developer. The specification becomes a "contract" for software development. Requests for changes in requirements after the specification is finalized will not be eliminated. But the customer should note that each after- the-fact change is an extension of software scope and therefore can increase cost, and/or protract the schedule.

Even with the best review procedures in place, a number of common specification problems persist. The specification is difficult to "test" in any meaningful way, and therefore inconsistency or omissions may pass unnoticed. During the review, changes to the specification may be recommended. It can be extremely difficult to assess the global impact of a change; that is, how a change in one function affects requirements for other functions. Modem software engineering environments incorporate CASE tools that have been developed to help solve these problems.

LECTURE # 16

2. Software Development Fundamentals
Technical Fundamentals

2.12 Design

⇒ **Management Aspect**

– ***Major design styles***
  • Object, structured, data-structured design

– ***Foundational design concepts***
  • Information hiding, abstraction, encapsulation, Inheritance, basic algorithms & data structures, ...

⇒ **Attributes of SW Design**

Software design is actually a multi step process that focuses on four distinct attributes of a program:

– Data structure,
– Software architecture,
– Interface representations, and procedural (algorithmic) detail.

The design process translates requirements into a representation of the software that can be assessed for quality before coding begins.

Like requirements, the design is documented and becomes part of the software configuration.

Design is the technical kernel of software engineering. During design, progressive refinements of data structure, architecture, interfaces, and procedural detail of software components are developed, reviewed, and documented.

Design results in representations of software that can be assessed for quality. A number of fundamental software design principles and concepts have been proposed over the past four decades.

Design principles guide the software engineer as the design process proceeds. Design concepts provide basic criteria for design quality.

a. Code generation

The design must be translated into a machine-readable form. The code generation step performs this task. If design is performed in a detailed manner, code generation can be accomplished mechanistically.

b. Testing

Once code has been generated, program testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested and on the functional externals; that is, conducting tests to uncover errors and ensure that defined input will produce actual results that agree with required results.

⇒ **Key Features of Design**

A number of fundamental software design principles and concepts have been proposed over the past four decades. Design principles guide the software engineer as the design process proceeds. Design concepts provide basic criteria for design quality.
Support

Support

Software will undoubtedly undergo change after it is delivered to the customer (a possible exception is embedded software). Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment (e.g. a change required because of a new operating system or peripheral device), or because the customer requires functional or performance enhancements.

Software support/maintenance reapplies each of the preceding phases to an existing program rather than a new one.

Modularity

Modularity (in both program and data) and the concept of abstraction enable the designer to simplify and reuse software components. Refinement provides a mechanism for representing successive layers of functional detail. Program and data structure contribute to an overall view of software architecture, while procedure provides the detail necessary for algorithm implementation.

Information hiding and functional independence provide heuristics for achieving effective modularity.
We try to solve the problem by rushing through the design process so that enough time will be left at the end of the project to uncover errors that were made because we rushed through the design process.

The moral is this: Don't rush through it! Design is worth the effort.

⇒ **Standard design approaches**

- Exception handling,
- Localization
- Portability
- Reuse
- Input/output
- Memory management,
- Performance

2.13 Construction

– ***Coding practices***
Naming, layout, documentation

– ***Data-related concepts***
Scope, persistence, binding time

– ***Data usage guidelines***
Bytes, arrays

– ***Use of construction tools***
Programming environment, group work support (documents, code), code libraries & generators

2.14 Software Configuration management

Software configuration management (SCM) takes care of changes in a software process. SCM identifies controls, audits, and reports modifications that occur during software development. SCM helps maintain the integrity of configurable items produced during software development. SCM is an integral part of Software Quality Assurance (SQA). SCM involves assessing the impact of the changes made during SQA activities and making decisions based on cost and benefit analysis.

SCM is used to establish and maintain integrity of software items and ensure that they can be traced easily. SCM helps define a library structure for storage and retrieval of software items.

- To ensure that project stays consistent over time: You need to:

  a) Evaluate proposed changes
  b) Track the changes
  c) Control the Version
  d) Check integrity of Source code, documents, plans, design
  e) Ensure quality

⇒ **Software Configuration Management Activities**

SCM is used to establish and maintain integrity of software items and ensure that they can be traced easily. Using SCM, you can define a library structure for storage and retrieval of software items. SCM needs to be performed at all phases in the SDLC of a software project. The various SCM activities are:

1. Identifying Objects
2. Controlling Versions
3. Controlling Changes
4. Auditing
5. Communicating Changes

1. Identifying Objects

The first activity in SCM involves identifying software configurable items (SCIs). SCI is an aggregation of software that is designated for configuration management. It is treated as a single entity in the configuration management process. For example, design documents, program code, test case, and custom requirement document are configurable items.

You can use the Item Traceability Matrix to identify SCIs at the end of each phase. A sample of Item Traceability Matrix is displayed in Table 1. In the table, you can see the different SCIs in different phases of the development process.

***Table* 1:** *Item Traceability Matrix*

Deliverable Name	Phase			
	Requirement Analysis	**Design**	**Construction**	**Testing**
Requirement Analysis Document	Requirement Analysis Document			
Design Document		Functional Specification Document and Program Specification Document		
Code			Code A Code B Code C	
Test Document				Unit Test Cases System Test Cases
Database Design Document		Database Design Specification		

To identify SCIs, you need to first breakdown the project deliverable to the SCI level. Each phase in the project has its own deliverables. To trace the deliverables, you need to map the SCIs to the phases in which they are delivered.

2. Controlling Versions

Version control combines procedures and tools to manage different versions of configuration objects that are created during software product development. To control versions, you can use Version Control Register. In Version Control Register, you enter the details of components, such as component identification numbers, their versions, and dates of validity. It is advisable to release a baseline after a version is released. Baseline is a specification or a product that is formally reviewed and agreed upon. This serves as the basis for further development. Baseline can be changed only through formal change control procedures. A baseline consists of a set of SCIs that are logically related to each other. Baselines are established when subsequent changes to the SCIs need to be controlled. Version control is essential so that everybody uses only the latest version. Any kind of version mismatch might result in rework.

LECTURE # 17

2. Software Development Fundamentals
Technical Fundamentals

2.14 Software Configuration management

b. Controlling Versions

Version control combines procedures and tools to manage different versions of configuration objects that are created during software product development. To control versions, you can use Version Control Register. In Version Control Register, you enter the details of components, such as component identification numbers, their versions, and dates of validity. It is advisable to release a baseline after a version is released. Baseline is a specification or a product that is formally reviewed and agreed upon. This serves as the basis for further development. Baseline can be changed only through formal change control procedures. A baseline consists of a set of SCIs that are logically related to each other. Baselines are established when subsequent changes to the SCIs need to be controlled. Version control is essential so that everybody uses only the latest version. Any kind of version mismatch might result in rework.

c. Controlling Changes

Uncontrolled change can lead to chaos. Change control combines human procedures and automated tools to provide a mechanism for controlling change. The purpose of change control is to monitor and control changes in order to baseline SCIs. There are various reasons that trigger changes. A problem report might call for a change. Similarly, suggestions or ideas from brainstorming sessions and feedback from clients can result in change. Modifications or addition to functionality and changes in environment can also cause changes. The Figure 1 explains the formal change control process using a flow chart.

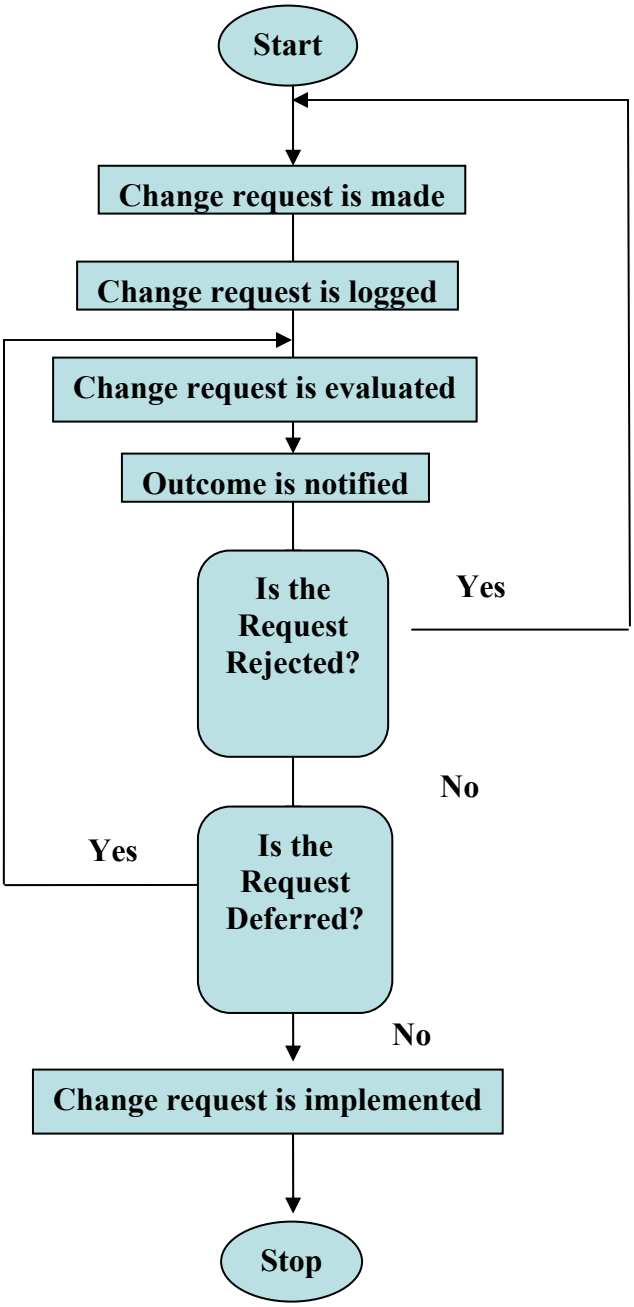


Figure 1: Formal Change Control Process

A request for change triggers that change control procedure. Then request is logged in the change request register. Next, the change request number is recorded in the change request evaluation plan. The request is evaluated and analyzed to check if the change is valid. Change request is also evaluated in

terms of the number of items affected and the effort involved in effecting the change. Finally, the possible outcome of the change request is communicated. The request for change is rejected, deferred, or approved. If the request for change is rejected, the requestor needs to log a fresh request. A deferred change request is evaluated at a later date while the change request that is approved is implemented.

There are tools that provide facilities to check in and check out so that the same version of the object is not updated more than once. The check-in and checkout facilities provide synchronization control. Synchronization control helps to ensure that parallel changes performed by two different people do not overwrite one another.

d. Auditing

Configuration audit is conducted formally by the SQA group in projects where SCM is a formal activity. The identification, version control, and change control tasks help the developer maintain order and decorum in an environment of change. However, the control mechanisms track a change only until an item is generated. FTRs and software configuration audits (SCA) are conducted to ensure that change is properly implemented. FTR verifies the technical correctness of the SCI that is subjected to change. SCA assesses those characteristics of an SCI that are not considered during FTRs. Audit verifies whether the changes specified in the request for change are made and additional modifications, if any, are also noted. Audits ensure that FTRs are conducted to check for technical correctness. Audits verify that changes made are highlighted in the SCI. The change date and change author are specified and the attributes of SCI reflect the change. The SCM procedures for noting, recording, and reporting change are also followed. Audits also ensure that related SCIs are updated.

e. Communicating Changes

Another task of SCM is communicating changes. This task ensures communication between different members in the project. It notes the activities performed, the time when they are performed, those involved in the activities, and those affected by the activities. In short, the task is all about status reporting. In a large project, there is a possibility of miscommunication among various people involved in the project. This is usually done using configuration status report shown in Table 11.7. The table contains the name of the SCI, the latest released version, the date of release, brief description of changes performed, and associated change request. Further details of the changes can be obtained from the associated change request. There can be instances where two developers may be trying to modify the same software configurable *item* with different and conflicting intentions. Absence of status reporting could result in incorrect decisions being taken or important decisions

not being communicated. At times, those who should be pointing out the serious side effects caused by a change are not aware of the implementation of the change. There are also instances of version mismatch when teams are unaware of the latest version to be followed. To avoid such hazards due to lack of communication among the project team, changes are communicated among team members. Therefore, status reporting provides information about each change to those who need to know. Software configuration management takes care of changes in a software process. SCM identifies controls, audits, and reports modifications that occur during software development. SCM helps maintain the integrity of configurable items produced during software development.

⇒ **Software Configuration Management Vs Software Maintenance**

SCM is an integral part of SQA. SCM involves assessing the impact of the changes made during SQA activities and making decisions based on cost and benefit analysis. SCM can be defined as the art of identifying, organizing, and controlling changes in a software project with the objective of minimizing mistakes. SCM is different from software maintenance. Software maintenance is required after the software is delivered to the client and is put into operation. As opposed to this, SCM is a set of tracking and controlling activities that begins when a software project begins and ends only when the software is taken out of operation.

⇒ **Baselines vs. Interim Versions**

SCM differentiates between baselines and interim versions. A baseline is a tested and certified version of a system. Baselines can be assigned version numbers such as 1.0, 2.0, 3.0, and so on. A baseline usually undergoes intensive testing. Interim versions, on the other hand, have version numbers, such as 1.1 or 1.2. The interim version is a temporary version. Interim versions have a short life and survive only during bug fixing, testing, or debugging. However, interim versions also have a unique version number or name. Baselines are more visible with the marketing team and the vendors than the interim versions. However, as part of SCM, all versions of changes are saved, clearly labeled, and archived. Archiving is the process of maintaining controlled copies of prior versions. Archiving helps in re-creating earlier versions in the event of data loss or data corruption.

⇒ **Effective Configuration Control**

Effective configuration control requires effective and well-defined organization. Any configuration control method must be based on the following four concepts:

A clearly defined configuration management authority must be established.
Configuration control standards, procedures and guidelines must be produced and distributed to the developers.

Configuration control cannot be effective without the necessary tools and facilities.
A configuration management plan must be developed at the beginning of the project.

The configuration management environment consists of the resources necessary for the implementation of the configuration control plan.

Configuration control tools, including:

- Automatic version control and
- Change control tools
- Monitoring, auditing and registration support utilities
- Storage facilities; a safe repository for all approved configuration items, including:
  - On-site storage for the day to day development process
  - Off-site storage for catastrophe recovery

⇒ **Guidelines for effective configuration management**

The following are some additional guidelines for effective configuration management. Some of these guidelines are equally applicable to other management support functions.

- Configuration management requires authority in order to be effective. This authority must be clearly delegated by the project manager to the responsible engineers. Any configuration management plan will become meaningless if the plan cannot be enforced.

- Blunt enforcement of any plan policy or standard is best avoided, whenever possible. One of the qualities of a good manager is the ability to apply policy with minimal enforcement. Whenever policies and standards are readily accepted by the developers, they are more willingly followed and there are fewer rejections of submitted material. This leads to a more efficient development process.

- Configuration management should be implemented from the start of a software project. Many of the formal documents issued during the initial concept phase are crucial for the requirements and design phases, and must be placed under configuration control.

- The early application of configuration management is especially important in rapid prototyping, spiral models, or other iterative development methodologies. These development approaches initially produce several

versions of each product. Many different versions can become an engineering nightmare without orderly configuration control.

- Occasionally some software configuration control activities may overlap with software quality assurance activities. In small projects, these two functions may be assigned to a single support engineer. Even in large projects, these two functions are sometimes performed by a single support group.

⇒ **Misapplication of Guidelines**

It should be noted that configuration management can be greatly exaggerated. The various configuration control activities are not an objective in themselves, they are a means.

- A typical example of the misapplication of configuration management (and misguided quality control), is a requirement to modify reused software to comply with current standards and procedures.

- Reused software is software developed previously in another project, and found suitable to be incorporated into the current project. In such cases it rarely makes sense to modify a complete and working product in order to make it comply with administrative standards intended to make it a complete and working product.

LECTURE # 18

2. Software Development Fundamentals
Technical Fundamentals

2.15 Quality Assurance Management

⇒ **Definition**

According to American Heritage Dictionary, quality is defined as "***an inherent or distinguishing characteristic or a property***".

The distinguishing characteristics of a software product are the ***cyclomatic complexity, cohesion, function points, and lines of code***. These characteristics of a software product define the quality of the product.

- The US DOD (1988) defines software quality rather simply as:

– The ability of a software product to satisfy its specified requirements.

- The British Standards Institution (1986) has stated that:

– "Quality is in the eye of the beholder, a matter, of the client's judgment."

⇒ **Quality Standards and Procedures**

The quality of software is said to be high if it meets the standards and procedures, defined for the product. Standards are criteria to which the products are compared. For example, there may be standards that govern the quality review process.

Documentation standard design standard and code standard are the three types of standards that software projects usually follow.

Documentation standard specifies the form and content for planning, control, and product documentation. Design standards provide rules and methods for translating the software requirements into software design. The design standards are specified in the form and content of the product design.

Unlike documentation standard, code standard defines the language in which code should be written. The standard clearly mentions the structures, style conventions, and rules for data structures and interfaces that will be implemented in the project.

Procedures are criteria to which the development and control processes are compared. Procedures are explicit steps followed in a process. Procedures need to be properly documented because they are needed for configuration management, nonconformance reporting, corrective action, testing, and formal inspections. Proper documentation of procedures is necessary because SQA activities rely on them for project compliance. Organizations normally enforce quality standards with the help of checklists, common error lists, and standards and guidelines.

⇒ **Concepts**

Product quality depends on its conformance to software requirements, development standards, and implicit requirements.

During software development, the quality of a product depends on the quality of the design.

The quality of product design, in turn, depends on how effectively the product designer captures the client requirements and specifications. At times, the client has some implicit requirements that are not captured in the requirements document. There are three things that guide requirements: want, desire, and wish. Usually, wants are captured explicitly in the requirements documents. However, if you manage to capture desires and wishes, the product becomes a great success. The product designer needs to state these implicit requirements clearly during analysis. The adherence to these implicit requirements is the key attribute that sets one product apart from another.

The product quality also depends on how strictly, and to what degree the developer adheres to design specifications. This is what the concept quality is meeting or exceeding our client's needs and requirements' means.

The product quality is said to be high if the product is manufactured according to design specifications.

⇒ **Quality Control**

Quality control is a series of review activities, such as:

- Inspections
- Reviews and
- Tests, used throughout the SDLC of the software product

The objective of quality control is to find problems as early as possible and fix them.

Quality control ensures that the software product meets the requirements defined at every stage in its development.

There is provision for feedback mechanism during quality control. Any slippage in meeting the requirements during the development process is communicated to the development team immediately.

Feedback ensures that errors or misses found during quality control are rectified as soon as they are detected.

During software development, quality control plays a valuable role by evaluating products against standards and specifications.

Quality control activities can be fully automated manual, or a combination of these. Quality control involves monitoring specific project results to determine if:

- They comply with relevant quality standards
- Identifying ways to eliminate causes of unsatisfactory results.

Project results include both:

- Product results, such as deliverables, and
- Project management results, such as cost and schedule performance.

⇒ **Quality Factors**

There are a number of factors that determine the quality of a software product. These factors can be measured either directly or indirectly. McCall (MCC77) and his colleagues proposed some software quality factors based on three most important aspects of a software product:

1. Product operation
2. Product revision and
3. Product transition

1. Product Operation Factors

The product operation factors determine the quality of software when a program is executed. Good quality software is not only correct and reliable but also delivers correct performance in all circumstances. Some of the factors of product operation are correctness, reliability, efficiency, integrity, and usability. You can look at the factor description in Table 1.

***Table* 1:** Product Operation Factors

Correctness	Accuracy of the program and the extent to which it fulfills design specifications
Reliability	Extent to which the program is secure and its ability to recover quickly from failure.

Efficiency	Performance of the program and its ability to perform tasks within a time frame
Integrity	Ability of the program to take care of security and the extent to which it can prevent unauthorized.
Usability	Ease with which a user can learn, operates, and uses the program.

2. **Product Revision Factors**
   Product revision factors focus on the ease of maintenance of the software product. Maintenance tasks could be either correcting faults in the original design or making improvements to adapt the functionality to changing environments. Product revision covers the following factors: maintainability, flexibility, and testability. These factors are described in Table 2.

***Table* 2:** Product Revision Factors

Maintainability	Ease with which a program is debugged.
Flexibility	Ease with which a program is modified.
Testability	Ease with which a program is tested.

3. **Product Transition Factors**
   The product transition factors determine the quality of programs that are designed for open systems. Here the focus is more on the portability and reusability of a software product. To help a system run on different platforms, certain parts of a system may be reused. The product transition factors are portability, reusability, interoperability, configurability, and expandability. These factors are described in Table 3.

***Table* 3:** Product Transition Factors

Portability	Efficiency with which a program runs on different platforms or operating systems
Reusability	Extent to which the program can be used in more than one program or system
Interoperability	Effort needed to transfer a program to another system
Configurability	Ability of the program to be installed at more than one location with different features at each location
Expandability	Ability of the program to support an increase in data and users

LECTURE # 19

2. Software Development Fundamentals
Technical Fundamentals

2.15 Quality Assurance Management

⇒ Software Quality Assurance Activities

SQA is the process of evaluating the quality of a product and enforcing adherence to software product standards and procedures. It is an umbrella activity that ensures conformance to standards and procedures throughout the SDLC of a software product. There are a large number of tasks involved in SQA activities. These include:

i. Formulating a quality management plan
ii. Applying software engineering techniques
iii. Conducting formal technical reviews
iv. Applying a multi-tiered testing strategy
v. Enforcing process adherence
vi. Controlling change
vii. Measuring impact of change
viii. Performing SQA audits
ix. Keeping records and reporting

i. Formulating a Quality Management Plan

One of the tasks of SQA is the formulation of a quality management plan. The quality management plan identifies the quality aspects of the software product to be developed. It helps in planning checkpoints for work products and the development process. It also tracks changes made to the development process based on the results of the checks. The quality management plan is tracked as a live plan throughout the SDLC.

ii. Applying Software Engineering

Application of software engineering techniques helps the software designer to achieve high quality specification. The designer gathers information using techniques such as interviews and FAST. Using the information gathered, the designer prepares project estimation with the help of techniques such as WBS, SLOC estimation, or FP estimation.

iii. Conducting Formal Technical Reviews

Formal technical review (FTR) in conducted to assess the quality and design of the prototype. It is a meeting with the technical staff to discuss the quality

requirements of a software product and its design quality. FTRs help in detecting errors at an early phase of development. This prevents errors from percolating down to the latter phases and resulting in rework.

iv. Applying a Multi-tiered Testing Strategy

Software testing is a critical task of SQA activity, which aims at error detection. Unit testing is the first level of testing. The subsequence levels of testing are integration testing and system level testing. There are various testing strategies followed by organization. At times, developers perform unit testing and integration testing with independence testing support. There are also occasions where testers perform functional testing and system level testing with developer support. Sometimes beta testing with selected clients is also conducted to test the product before it is finally released.

v. Enforcing Process Adherence

This task of SQA emphasizes the need for process adherence during product development. In addition, the development process should also adhere to procedures defined for product development. Therefore, this is a combination of two tasks, product evaluation and process monitoring.

vi. Product Evaluation

Product evaluation ensures that the standards laid down for a project are followed. During product evaluation, the compliance of the software product to the existing standards is verified. Initially, SQA activities are conducted to monitor the standards and procedures of the project. Product evaluation ensures that the software product reflects the requirements identified in the project management plan.

vii. Process Monitoring

Process monitoring ensures that appropriate steps to follow the product development procedures are carried out. SQA monitors processes by comparing the actual steps carried out with the steps in the documented procedures.

Product evaluation and process monitoring ensure that the development and control processes described in the project management plan are correctly carried out. These tasks ensure that the project-re1ated procedures and standards are followed. They also ensure that products and processes conform to standards and procedures. Audits ensure that product evaluation and process monitoring are performed.

viii. Controlling Change

This task combines human procedures and automated tools to provide a mechanism for change control. The change control mechanism ensures software quality by formalizing requests for change, evaluating the nature of change, and controlling the impact of change. Change control mechanism is implemented during the development and maintenance stages.

ix. Measuring Impact of Change

Change is inevitable in the SDLC. However, the change needs to be measured and monitored. Changes in the product or process are measured using software quality metrics. Software qua1ity metrics helps in estimating the cost and resource requirements of a project. To control software quality; it is essential to measure quality and then compare it with established standards. Software qua1ity metrics are used to evaluate the effectiveness of techniques and tools, the productivity of development activities and the qua1ity of products. Metrics enables mangers and developers to monitor the activities and proposed changes throughout the SDLC and initiate corrective actions.

x. Performing SQA Audits

SQA audits scrutinize the software development process by comparing it to established processes. This ensures that proper control is maintained over the documents required during SDLC. Audits also ensure that the status of an activity performed by the developer is reflected in the status report of the developer.

xi. Keeping Records and Reporting

Keeping records and reporting ensure the collection and circulation of information relevant to SQA. The results of reviews, audits, change control, testing, and other SQA activities are reported and compiled for future reference.

⇒ **Software Review**

Software review is an effective way of filtering errors in a software product. Typically, an error found after the product release costs 50 times as much to correct as one detected during the design phase.

This means the cost of fixing a defect rises dramatically if it is found late in the development life cycle of the product. Therefore, you need to filter errors as early as possible.

Software review is used as a filter at various points of software development. Reviews conducted at each of these phases, analysis, design, coding, and testing reveal areas of improvement in the product.

Reviews also indicate those areas that do not need any improvement. You can use software reviews to achieve consistency and uniformity across products. Reviews also make the task of product creation more manageable. Some of the most common software review techniques, practiced across software organizations include:

a) Inspection
b) Walkthrough
c) Formal technical reviews

a) Inspections

Inspections improve the reliability, availability, and maintainability of a software product. Anything readable that is produced during software development can be inspected. The readable material can be requirements specifications, design documents and models, test plans, system documentation, and user aids. Group inspections enable team members to exchange knowledge and ideas during an inspection session.

Inspections can be combined with structured, systematic testing to provide a powerful tool for creating defect-free programs. The inspection activity follows a specified process and the participants play well-defined roles. An inspection team consists of three to eight members who play the roles of moderator, author, reader, recorder, and inspector.

Moderator leads the inspection, schedules meetings, controls meetings, reports inspection results, and follows up on rework issues. Author creates or maintains the work product being inspected. Reader describes the sections of the work product to the team as they proceed through inspection. Recorder classifies and records defects and issues raised during the inspection. The moderator might perform this role in a small inspection team. Inspector finds errors in the product. All participants play the role of inspectors. However, good inspectors are those who have created the specification for the work product being inspected. For example, the designer can act as an inspector during code inspection while a quality assurance representative can act as standard enforcer. It also helps to have a client representative participate in requirements specification inspections.

b) Walkthroughs

The term walkthrough refers to a group activity in which the developer of the product guides the progress of the review. Walkthroughs are less rigorous than either formal inspections or peer reviews in which the developer plays a more passive role. Normally walkthroughs turn into a presentation by the author. The focus of finding errors is diluted. Such misadventures make walkthroughs usually less successful at detecting bugs than the more formal review methods.

Two useful walkthrough approaches adopted worldwide are group reviews with individual preparation and individual peer desk-checks. Group reviews are not very rigorous like inspections. The group reviews involve many of the same activities and roles, such as individual preparation and the use of a moderator and a recorder. Usually the overview meeting and the follow-up steps are skipped and checklists are used sparingly. At the end, the readers paraphrase their interpretation of what a program is doing.

Individual peer desk-checks are quite cost-effective. Only one person besides the author examines the material. This approach can be more effective if there are individuals who are extremely good at finding defects on their own.

Tip:
If someone consistently finds most of the group-identified defects during the individual preparation step, such a person is fit to perform individual peer desk-checks.

c) Formal Technical Reviews

One of the most common, yet important, software quality assurance activity performed is FTR. This activity is performed to check errors in logic, function, or implementation for any representation of the software.

Using FTR, you can verify whether or not the software product adheres to the defined standards. FTR is also conducted to check for consistency in the software product and audit the manageability of the software product. It includes activities such as walkthrough and inspection.

FTR focuses on specific parts of the software product, such as the requirements components, detailed design module, and the source code listing for a module.

FTR also concentrates on the entire product. The participants in FTR are the developer, the project leader, and all the reviewers.

At the end of the review meeting, the issues recorded are formalized into review issues list. The minutes of the meeting are summarized into a review summary report as displayed in table 4.

Technical Review Report		
Review Identification		
Project Name	Current Phase	Review Number
Product Identification		
Item Reviewed	Version No.	Developer
Brief Description		
Documents Referred (along with versions, if any)		
Start Name	End Time	
Review Material Attached: Yes / No		
Participants	Role	Preparation Time

Total Review Time (No. of Participants * Actual duration of meeting)		
Total Review Person Hours		
Product Appraisal		
Accepted as is ( )	With Minor Modification ( )	
Not Accepted Major Revision ( )	Minor Revision ( )	
Review Not Completed (Explanation as follows)		
Supplementary Material Attached		
Issue Lists:	Others	
Revise and Reschedule (if required)	Target Date for Next Review	
Verification of Defect Closure		
Responsibility of:	Planned Date of Verification	
Recommended for Release	Signature:	Date:

Table 4: Technical Review Report

Tip:

To prepare realistic schedule, you can collect data from past projects. Speak to related project managers regarding time estimation for reviews, the review capacity of the reviewer, and estimates the review effort accordingly.

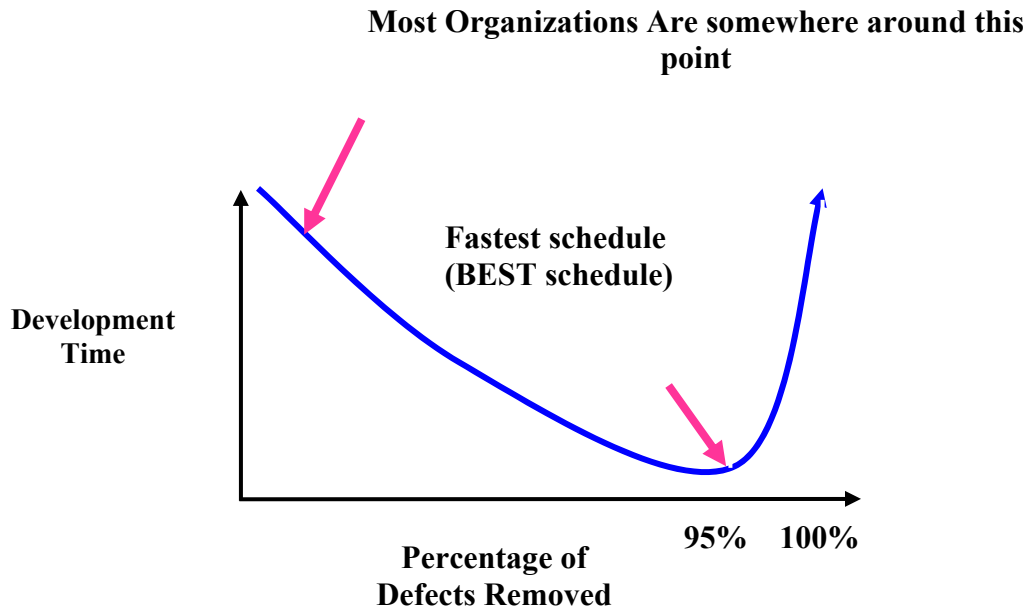


Figure 1: Quality Assurance

LECTURE # 20

3. Processes

3.1 Software Process

When you build a product or system, it's important to go through a series of predictable steps – a road map that helps you create a timely, high-quality result, The road map that you follow is called a 'software ***process'*** .

Software engineers and their managers adapt the process to their needs and then follow it. In addition, the people who have ties defined by the process requested the software play a role in the software process.

It provides stability, control, and organization to an activity that can, if left uncontrolled, become quite chaotic.

At a detailed level, the process that you adopt depends on the software you're building. One process might be appropriate for creating software for an aircraft avionics system, while an entirely different process would be indicated for the creation of a web site.

From the point of view of a software engineer, the work products are the programs, documents and data produces as a consequence of the software engineering activities defined by the process.

A number of software process assessment mechanisms enable organizations to determine the “maturity” of a software process. However, the quality, timeliness and long-term viability of the product you build are the best indicators of the efficacy of the process that you use.

A ***common process framework*** is established by defining a small number of framework activities that are applicable to all software projects, regardless of their size or complexity. A number of *task* sets-each a collection of software engineering work tasks, project milestones, work products, and quality assurance points enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team. Finally, umbrella activities-such as software quality assurance, software configuration management, and measurement - overlay the process model. Umbrella activities are independent of anyone framework activity and occur through- out the process.

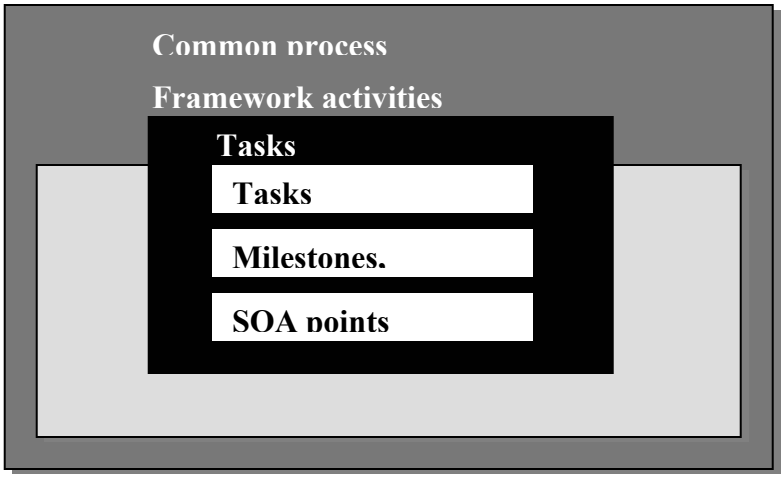


Figure 1: The software Process

⇒ **Software Process Vs Software Engineering**

But what exactly is a software process from a technical point of view? A *software process* is a framework for the tasks that are required to build high-quality software. Is *process* synonymous with software engineering? The answer is "yes" and "no." A software process defines the approach that is taken as software is engineered. But software engineering also encompasses technologies that populate the process-technical methods and automated tools.

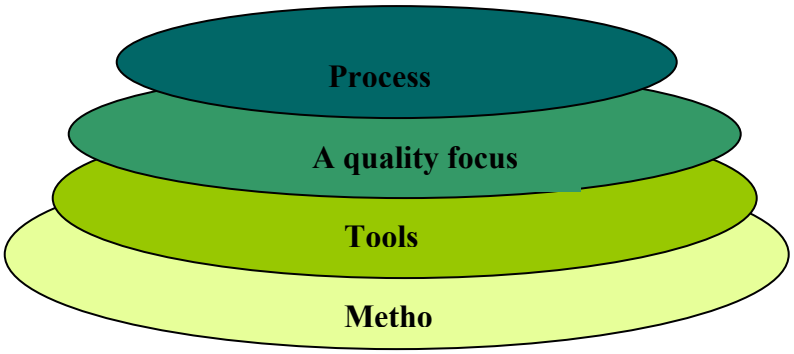
More important, software engineering is performed by creative, knowledgeable people who should work within a defined and mature software process that is appropriate for the products they build, and the demands of their marketplace.

⇒ **Software Engineering**

Software engineering is a layered technology. Referring to Figure 2, any engineering approach (including software engineering) must rest on an organizational commitment to quality. Total quality management and similar philosophies foster a continuous process improvement culture, and this culture ultimately leads to the development of increasingly more mature approaches to

software engineering. The bedrock that supports software engineering is a quality focus.

Figure 2: Software engineering layers



The foundation for software engineering is the process layer. Software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software. Process defines a framework for a set of key process areas that must be established for effective delivery of software engineering technology.

The key process areas form the basis for management control of software projects and establish the context in which technical methods are applied, work products (models, documents, data, reports, forms, etc.) are produced, milestones are established, quality is ensured, and change is properly managed.

Software engineering methods provide the technical how-to's for building software. Methods encompass a broad array of tasks that include requirements analysis, design, program construction testing and support. Software engineering methods rely on a set of basic principles that govern each area of the technology and include modeling and other descriptive techniques.

Software engineering tools provide automated or semi-automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another, a stem for the support of software development called computer-aided software engineering, is established. CASE combines software, hardware, and a software engineering database (a repository containing important information about analysis, design, program construction and testing) to create software engineering environment analogous to CAD/CAE (computer-aided design engineering) for hardware.

3.2 PM Process Groups

Project management processes can be organized into five groups of one or more processes each:

1. **Initiating** processes—authorizing the project or phase

2. **Planning** processes—defining and refining objectives and selecting the best of the alternative courses of action to attain the objectives that the project was undertaken to address

3. **Executing** processes—coordinating people and other resources to carry out the plan

4. **Controlling** processes—ensuring that project objectives are met by monitoring and measuring progress regularly to identify variances from plan so that corrective action can be taken when necessary

5. **Closing** processes—formalizing acceptance of the project or phase and bringing it to an orderly end

Each process is described by:

- Inputs
- Tools & Techniques
- Outputs

3.3 PM Process Links

The process groups are linked by the results they produce—the result or outcome of one often becomes an input to another.

Among the central process groups, the links are iterated—planning provides executing with a documented project plan early on, and then provides documented updates to the plan as the project progresses. These connections are illustrated in **Figure 3.**

The project management process groups are not discrete, one-time events; they are overlapping activities that occur at varying levels of intensity throughout each phase of the project. These process groups overlap and vary within a phase. **Figure 4** illustrates how the process groups overlap and vary within a phase.

3.4 PM Phase interactions

Finally, the process group interactions also cross phases such that closing one phase provides an input to initiating the next. For example, closing a design phase requires customer acceptance of the design document. Simultaneously, the design document defines the product description for the ensuing implementation phase. This interaction is illustrated in **Figure 5**. Repeating the initiation processes at the start of each phase helps to keep the project focused on the business need that it was undertaken to address. It should also help ensure that the project is halted if the business need no longer exists, or if the project is unlikely to satisfy that need.

It is important to note that the actual inputs and outputs of the processes depend upon the phase in which they are carried out. Although **Figure 5** is drawn with discrete phases and discrete processes, in an actual project there will be many overlaps. The planning process, for example, must not only provide details of the work to be done to bring the current phase of the project to successful completion, but must also provide some preliminary description of work to be done in later phases. This progressive detailing of the project plan is often called *rolling wave planning*; *indicating that planning is an iterative and ongoing process*.

Involving stakeholders in the project phases generally improves the probability of satisfying customer requirements and realizes the *buy-in* or shared ownership of the project by the stakeholders, which is often critical to project success.

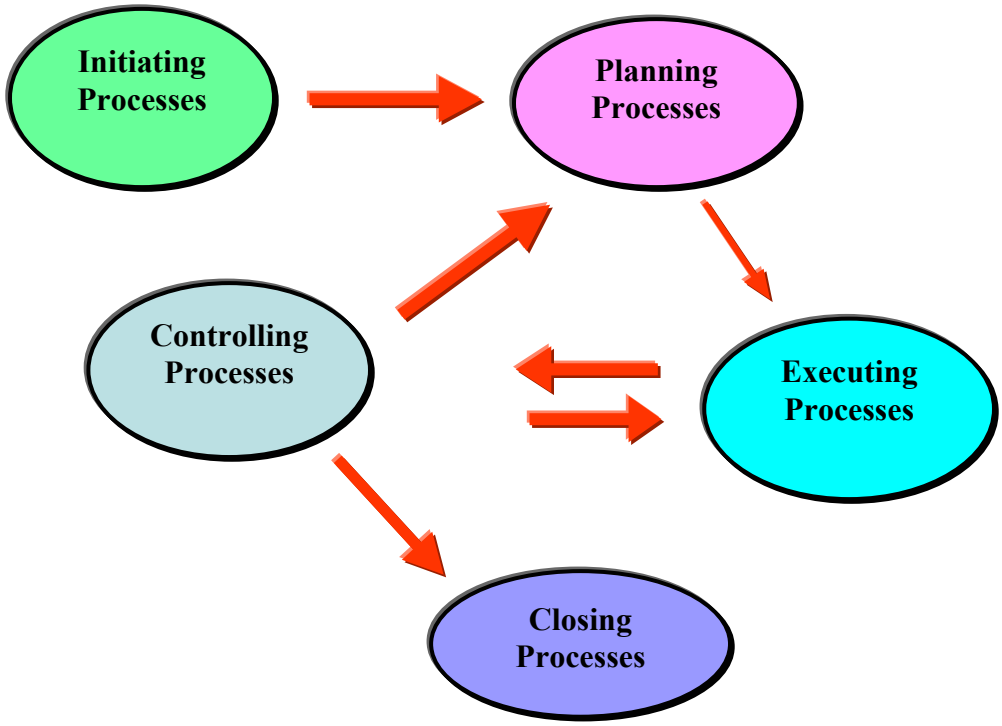


Figure 3: Links among Process Groups in a phase

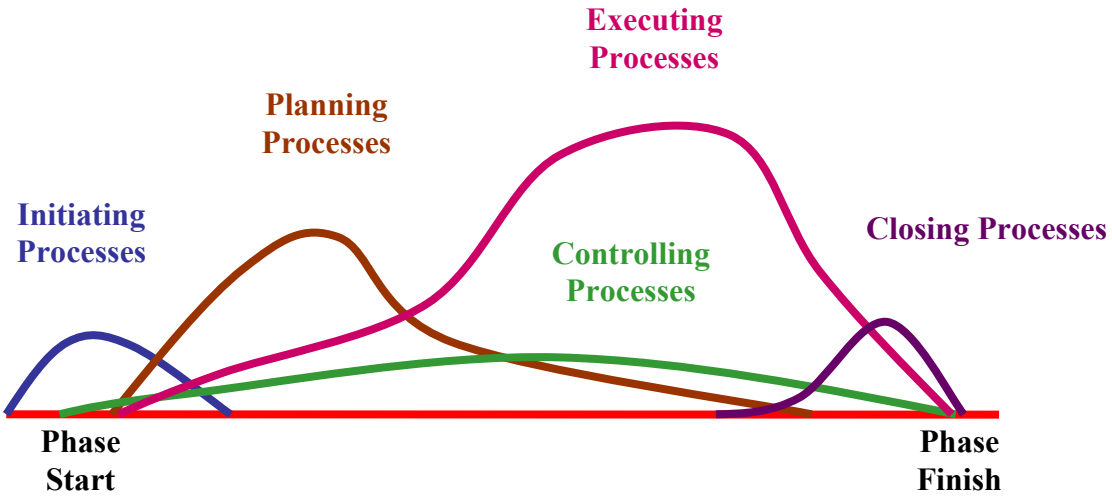


Figure 4: Overlap of Process Groups in a Phase

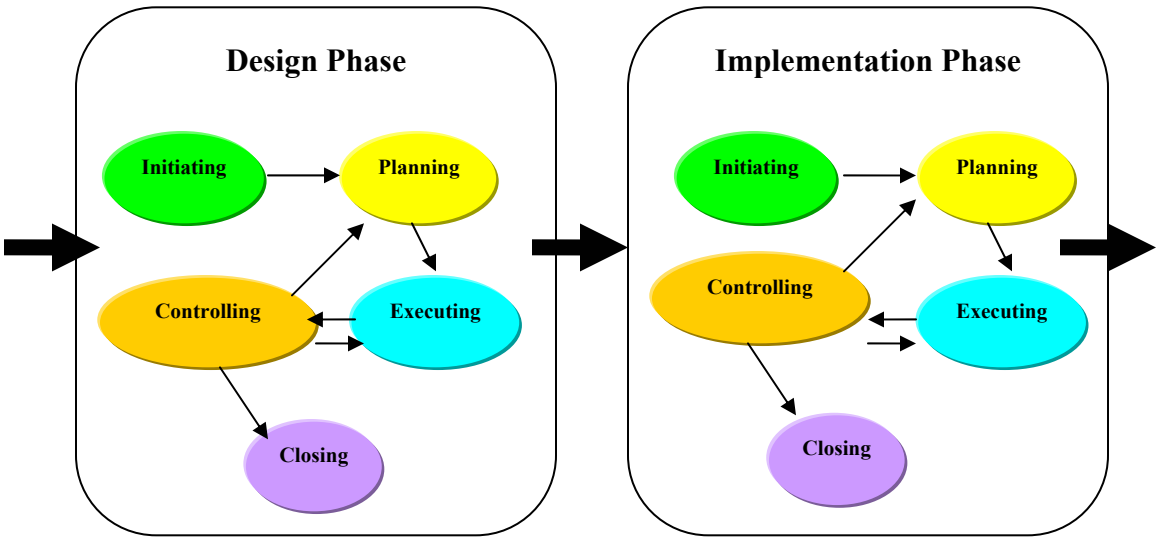


Figure 5: Interaction between phases

LECTURE # 21

3. Processes

3.5 Initiating Process

⇒ **Inputs**
- Product Description
- Strategic Plan
- Selection Criteria
- Historical Information

⇒ **Outputs**
- Project Charter
- Project Manager assignments
- Constraints
- Assumptions

⇒ **Tools and Techniques**
- Project selection methods
- Expert judgment

The tasks performed for project initiation are mentioned below:

- ***Requirement gathering*:** The first task is to gather the customer requirements. Customer requirements may be spoken or unspoken. Therefore, the challenge for the project manager is to elicit the requirements in such a way that both the spoken and unspoken customer needs and wants are gathered. After collecting the required information, you need to translate the customer requirements into technical specifications for the software project.

- ***Scope determination***: The scope of a software project can be defined as the combination of the software product arid services to be delivered to the customer. You carry out the scope determination exercise to define the scope of the software project. The scope determination exercise enables you to refine and understand the customer requirements. You can refine the scope definition further by breaking down each deliverable into smaller and more manageable activities. The scope determination exercise also helps you identify the technology for creating the software product.

- ***Resource allocation***: During project initiation, you identify the resources required and allocate them to the software project. The resources identified may be people, reusable software components, and hardware or software tools. You allocate the resource to the software project on the basis of the activities defined in the scope

determination exercise. While allocating appropriate resources for a software project, you also need to calculate the cost of each resource. The cost of a resource is calculated according to the duration of the resource in the software project. Estimating the cost of resources also helps you prepare a budget for the software project.

- ***Initial project plan:*** Another exercise that you carry out during project initiation is the creation of a rough project plan. This plan is a draft version and carries only the primitive project plan features. This project plan carries the initial risk analysis of the software project, the initial start and end dates, the duration of the activities in the project, and the sequencing of these activities.

3.6 Planning Process

Devising and maintaining a workable scheme to accomplish the business need that the project was undertaken to address

Planning is setting the direction for something -- some system -- and then guiding the system to follow the direction.

The basic planning process typically includes similar nature of activities carried out in similar sequence.

The phases are carried out carefully or -- in some cases -- intuitively, for example, when planning a very small, straightforward effort the complexity of the various phases (and their duplication throughout the system) depend on the scope of the system.

For example, in a large corporation, the following phases would be carried out in the corporate offices:

- **In each division**
- **In each department**
- **In each group, etc.**

Planning typically includes use of the following basic terms

- **Goals:** Goals are specific accomplishments that must be accomplished in total, or in some combination, in order to achieve some larger, overall result preferred from the system, for example, the mission of an organization. (goals are outputs from the system.)

- **Strategies or Activities:** These are the methods or processes required in total, or in some combination, to achieve the goals. (strategies are processes in the system.)

- **Objectives:** Objectives are specific accomplishments that must be accomplished in total, or in some combination, to achieve the goals in the plan. Objectives are usually "milestones" along the way when implementing the strategies

- **Tasks:** Particularly in small organizations, people are assigned various tasks required to implement the plan. If the scope of the plan is very small, tasks and activities are often essentially the same.

- **Resources (and Budgets):** Resources include the people, materials, technologies, money, etc., required to implement the strategies or processes. The costs of these resources are often depicted in the form of a budget. (Going back to our reference to systems, resources are input to the system.)

  Goals and Objectives Should Be **SMARTER**

  SMARTER is an acronym, that is, a word composed by joining letters from different words in a phrase or set of words. In this case, a **SMARTER** goal or objective is:

  Specific:
  For example, it's difficult to know what someone should be doing if they are to pursue the goal to "work harder". It's easier to recognize "Write a paper".

  Measurable:
  It's difficult to know what the scope of "Writing a paper" really is. It's easier to appreciate that effort if the goal is "Write a 30-page paper".

  Acceptable:
  If I'm to take responsibility for pursuit of a goal, the goal should be acceptable to me. For example, I'm not likely to follow the directions of someone telling me to write a 30-page paper when I also have to five other papers to write.

  However, if you involve me in setting the goal so I can change my other commitments or modify the goal, I'm much more likely to accept pursuit of the goal as well.

  Realistic: Even if I do accept responsibility to pursue a goal that is specific and measurable, the goal won't be useful to me or others if, for example, the goal is to "Write a 30-page paper in the next 10 seconds".

  Time frame: It may mean more to others if I commit to a realistic goal to "Write a 30-page paper in one week". However, it'll mean more to others (particularly if they are planning to help me or guide me to reach the goal) if I specify that I will write one page a day for 30 days, rather than including the possibility that I will write all 30 pages in last day of the 30-day period.

Extending: The goal should stretch the performer's capabilities. For example, I might be more interested in writing a 30-page paper if the topic of the paper or the way that I write it will extend my capabilities.

Rewarding: I'm more inclined to write the paper if the paper will contribute to an effort in such a way that I might be rewarded for my effort.

LECTURE # 22

3. Processes

⇒ **Planning Process Tasks**

1. **Scope Planning**
2. **Scope Definition**
3. **Activity Definition**
4. **Activity Sequencing**
5. **Activity Duration Estimating**
6. **Resource Planning**
7. **Cost Estimating**
8. **Cost Budgeting Risk Planning**
9. **Schedule Development**
10. **Quality Planning**
11. **Communications Planning**
12. **Organization Planning**
13. **Staff Acquisition**
14. **Procurement Planning**
15. **Project Plan Development**

1) **Scope Planning** — developing a written scope statement as the basis for future project decisions

2) **Scope Definition**—subdividing the major project deliverables into smaller, more manageable components

3) **Activity Definition**—identifying the specific activities that must be performed to produce the various project deliverables

4) **Activity Sequencing**—identifying and documenting interactivity dependencies

5) **Activity Duration Estimating**—estimating the number of work periods that will be needed to complete individual activities.

6) **Resource Planning**—determining what resources (people, equipment, materials, etc.) and what quantities of each should be used to perform project activities.

7) **Cost Estimating**—developing an approximation (estimate) of the costs of the resources required to complete project activities.

8) **Cost Budgeting**—allocating the overall cost estimates to individual work packages.

9) **Schedule Development** —analyzing activity sequences, activity durations, and resource requirements to create the project schedule.

10) **Quality Planning**—identifying which quality standards are relevant to the project and determining how to satisfy them.

11) **Communications Planning**—determining the information and communications needs of the stakeholders: who needs what information, when will they need it, and how will it be given to them.

12) **Organizational Planning**—identifying, documenting, and assigning project roles, responsibilities, and reporting relationships.

13) **Staff Acquisition**—getting the human resources needed assigned to and working on the project.

14) **Procurement Planning**—determining what to procure, how much to procure, and when.

15) **Project Plan Development** —taking the results of other planning processes and putting them into a consistent, coherent document.

3.7 Executing Process

⇒ **Executing Process Tasks**

1. **Project Plan Execution**—carrying out the project plan by performing the activities included therein.

2. **Quality Assurance**—evaluating overall project performance on a regular basis to provide confidence that the project will satisfy the relevant quality standards

3. **Team Development** —developing individual and group skills/competencies to enhance project performance

4. **Information Distribution**—making needed information available to project stakeholders in a timely manner.

5. **Solicitation** —obtaining quotations, bids, offers, or proposals as appropriate.

6. **Source Selection** —choosing from among potential sellers.

7. **Contract Administration** —managing the relationship with the seller.

3.8 Controlling Process

Ensuring that project objectives are met by monitoring and measuring progress and taking corrective measures when necessary

Project performance must be monitored and measured regularly to identify variances from the plan. Variances are fed into the control processes in the various knowledge areas. To the extent that significant variances are observed (i.e., those that jeopardize the project objectives), adjustments to the plan are made by repeating the appropriate project planning processes. For example, a missed activity finish date may require adjustments to the current staffing plan, reliance on overtime, or tradeoffs between budget and schedule objectives. Controlling also includes taking preventive action in anticipation of possible problems.

⇒ **Controlling Process Tasks**

1. **Integrated Change Control**
2. **Scope Verification**
3. **Scope Change Control**
4. **Schedule Control**
5. **Cost Control**
6. **Quality Control**
7. **Performance Reporting**
8. **Risk Response Control**

The controlling process group contains core processes and facilitating processes.

1. **Integrated Change Control —**coordinating changes across the entire project.

2. **Scope Verification**—formalizing acceptance of the project scope.

3. **Scope Change Control**—controlling changes to project scope.

4. **Schedule Control**—controlling changes to the project schedule.

5. **Cost Control**—controlling changes to the project budget.

6. **Quality Control**—monitoring specific project results to determine if they comply with relevant quality standards and identifying ways to eliminate causes of unsatisfactory performance.

7. **Performance Reporting**—collecting and disseminating performance information. This includes status reporting, progress measurement, and

forecasting.

8. **Risk Response Control**—keeping track of identified risks, monitoring residual risks and identifying new risks, ensuring the execution of risk plans, and evaluating their effectiveness in reducing risk.

3.9 Closing Process

Formalizing acceptance of the project or phase and bringing it to an orderly end.

⇒ **Closing Process Tasks**

- **Contract Closeout** —completion and settlement of the contract, including resolution of any open items.

- **Administrative Closure**—generating, gathering, and disseminating information to formalize phase or project completion, including evaluating the project and compiling lessons learned for use in planning future projects or phases.

LECTURE # 23

4. PLANNING

Planning is one of the most important management activities and is an ongoing effort throughout the life of the project. Software project management begins with a set of activities that are collectively called Project Planning.

- Preliminary planning starts on day one even in the pre-project phase
- It should not be conducted "in secret"
- It needs buy-in and approval

4.1 Project Planning Objectives

The objective of software project planning is to provide a framework that enables the manager to make reasonable estimates of:

- **Resources**
- **Cost, and**
- **Schedule**

These estimates are made within a limited time frame at the beginning of a software project and should be updated regularly as the project progresses.
In addition, estimates should attempt to define best case and worst-case scenarios so that project outcomes can be bounded.

Planning is one of the most important management activities and is an ongoing effort throughout the life of the project.

Software project management begins with a set of activities that are collectively called Project Planning.

The software project planner must estimate following things before a project begins:

1. **How much will it cost?**
2. **How long will it take?**
3. **How many people will it take?**
4. **What might go wrong?**

4.2 Project Planning - Definition

- What is it? Software project planning involves estimation - your attempt to determine how much money, how much effort, how many resources, and how much time it will take to build a specific software-based system or product.
- **Who does it?** Software Project Managers- Using information solicited from customers and software engineers and software metrics data collected from past projects. It is advisable to generate your estimates using at least two different methods (as a cross check). Problem complexity and risk are considered before a final estimate is made.

- **What is the work product?** A Simple table *delineating* the tasks to be performed, the functions to be implemented and the cost, effort and time involved for each is generated. A list of required project resources is also produced.

- **How do I ensure that I've done it right?** That's hard, because you won't really know until the project has been completed. However, if you have experience and follow a systematic approach, generate estimates using solid historical data, create estimation, data points using at least two different methods, and factors in complexity and risk. You can feel confident that you've done a right job to achieve the targets**.**

4.3 Project Planning: Key Tasks

1. **Set goal and scope**
2. **Select lifecycle**
3. **Set organization team form**
4. **Start team selection**
5. **Determine risks**
6. **Create WBS**
7. **Identify tasks**
8. **Estimate size**
9. **Estimate effort**
10. **Identify task dependencies**
11. **Assign resources**
12. **Schedule work**

4.4 Project Management Process

The software development management section, which describes the organization and resources that will be used to develop the product, should always be included. The management section discusses how the facilities will be organized to support the development effort. This is one of the sections that provide much of the detail needed to prepare the heart of the development plan, namely the development schedule. The schedule provides answers to two basic planning questions: *what* and *when,* while much of the remaining sections discuss *how*.

The discussion in the *how* sections provides information on how the project will be organized, how risks will be handled, how reviews will be conducted, how standards will be applied, what development methodologies will be used, and how the product will be tested.

It is usually best to leave the completion of the schedule section for last. The schedule, being dependent on most of the other sections, is the most sensitive part of the development plan. After a first draft of the development plan is ready, an initial development schedule can then be prepared. As we shall see, the schedule will then be further refined as the development plan goes through progressive iteration.

Why is the system being developed? The answer to this question enables all parties to assess the validity of business reasons for the software work. Stated in another way, does the business purpose justify the expenditure of people, time and money?

What will be done, by when? The answers to these questions help the team to establish a project schedule by identifying key project tasks and the milestones that are required by the customer.

Who is responsible for a function? Earlier in this chapter, we noted that the role and responsibility of each member of the software team must be defined. The answer to this question helps accomplish this.

Where they are organizationally located? Not all roles and responsibilities reside within the software team itself. The customer, users, and other stake" holders also have responsibilities.

How will the job be done technically and managerially? Once product scope is established, a management and technical strategy for the project-must be defined.

How much of each resource is needed? The answer to this question is derived by developing estimates based on answers to earlier questions.

The definition phase focuses on *what.* That is, during definition, the software engineer attempts to identify what information is to be processed, what function and performance are desired, what system behavior can be expected, what interfaces are to be established, what design constraints exist, and what validation criteria are required to define a successful system. The key requirements of the system and the software are identified. Although the methods applied during the definition phase will vary depending on the software engineering paradigm (or combination of paradigms) that is applied, three major tasks will occur in some form: system or information engineering, software project planning and requirements analysis.

The development phase focuses on *how.* That is, during development a software engineer attempts to define how data are to be structured, how function is to be implemented within a software architecture, how procedural details are to be

implemented, how interfaces are to be characterized, how the design will be translated into a programming language (or nonprocedural language), and how testing will be performed. The methods applied during the development phase will vary, but three specific technical tasks should always occur: software design, code generation, and software testing.

The support Phase focuses on *Change* associated with error correction, adaptations required as the software's environment evolves, and changes due to enhancements brought about by changing customer requirements. The support phase reapplies the steps of the definition and development phases but does so in the context of existing software. Four types of change are encountered during the support phase:

1. **Correction**
   Even with the best quality assurance activities, it is likely that the customer will uncover defects in the software. *Corrective maintenance* changes the software to correct defects.

2. **Adaptation**
   Over time, the original environment (e.g., CPU, operating system, business rules, external product characteristics) for which the software was developed is likely to change. *Adaptive maintenance* results in modification I the software to accommodate changes to its external environment.

3. **Enhancement**
   As software is used, the customer/user will recognize additional functions that will provide benefit. *Perfective maintenance* extends the software beyond its original functional requirements.

4. **Prevention**
   Computer software deteriorates due to change, and because (this, *preventive maintenance,* often called *software reengineering,* must be conducted to enable the software to serve the needs of its end users, in essence, preventive maintenance makes changes to computer programs so that they ca be more easily corrected, adapted, and enhanced.

   In addition to these support activities, the users of software require continuing support. In-house technical assistants, telephone-help desks, and application-specific Web sites are often implemented as part of the support phase.

4.5 Planning Puzzle

Planning is one of the most important management activities and includes the preparation of good estimates, the maintenance of the development schedules and the efficient assignment of personnel.

i. Scope Planning

Scope planning is the process of progressively elaborating and documenting the project work (project scope) that produces the product of the project.

ii. **Risk Planning**
Risk management planning is the process of deciding how to approach and plan the risk management activities for a project. It is important to plan for the risk management processes that follow to ensure that the level, type, and visibility of risk management are commensurate with both the risk and importance of the project to the Organization.

iii. **Schedule Development**
Analyzing activity sequences, activity durations, and resource requirements to create the project schedule.

iv. **Cost Estimating**
Developing an approximation (estimate) of the costs of the resources required to complete project activities

v. **Project Control**
Project Control may be considered to be one of the continuous objectives for the project manager. As such he is responsible for taking remedial actions, within the defined terms of reference, to correct potential problems or taking risk avoidance measures. The prime objective is to protect the integrity of the project at all times.

4.6 Primary Planning Steps

Once the scope of the project has been defined in the Terms of Reference, the project enters the detailed planning phase. This involves the creation of a:

1. Project Plan (outlining the activities, tasks, dependencies and timeframes)
2. Resource Plan (listing the labor, equipment and materials required)
3. Financial Plan (identifying the labor, equipment and materials costs)
4. Quality Plan (providing quality targets, assurance and control measures)
5. Risk Plan (highlighting potential risks and actions taken to mitigate them)
6. Acceptance Plan (listing the criteria to be met to gain customer acceptance)
7. Communications Plan (listing the information needed to inform stakeholders)
8. Procurement Plan (identifying products to be sourced from external suppliers).

At this point the project has been planned in detail and is ready to be executed

By this stage, the benefits and costs of the project have been clearly documented, the objectives and scope have been defined, the project team has been appointed and a formal project office environment established. It is now time to undertake detailed planning to ensure that the activities performed in the execution phase of the project are properly sequenced, resourced, executed and controlled.

1. Develop Project Plan

The first step is to document the Project Plan. A 'Work Breakdown Structure' (WBS) is identified, which includes a hierarchical set of phases, activities and tasks to be undertaken on the project. After the WBS has been agreed, an assessment of the effort required to undertake the activities and tasks is made. The activities and tasks are sequenced, resources are allocated and a detailed project schedule is formed. This project schedule will become the primary tool for the Project Manager to assess the progress of the project.

2. Develop Resource Plan

Immediately after the Project Plan is formed, it is necessary to allocate the resources required to undertake each of the activities and tasks within the Project Plan. Although general groups of resources may have already been allocated to the Project Plan, a detailed resource assessment is required to identify the:

- Types of resources (labor, equipment and materials)
- Total quantities of each resource type
- Roles, responsibilities and skill-sets of all human resources
- Items, purposes and specifications of all equipment resource
- Items and quantities of material resource

A schedule is assembled for each type of resource so that the Project Manager can assess the resource allocation at each stage in the project.

3. Develop Financial Plan

Similar to the Resource Plan, a Financial Plan is prepared to identify the quantity of money required for each stage in the project. The total cost of labor, equipment and materials is quantified and an expense schedule is defined which provides the Project Manager with an understanding of the forecast spending vs. the actual spending throughout the project. Preparing a detailed Financial Plan is extremely important as the project's success will depend on whether or not it is delivered within the 'time, cost and quality' estimates for this project.

4. Develop Quality Plan

Meeting the quality expectations of the customer is critical to the success of the project. To ensure that the quality expectations are clearly defined and can reasonably be achieved, a Quality Plan is documented. The Quality Plan:

- Defines what quality means in terms of this project

- Lists clear and unambiguous quality targets for each deliverable. Each quality target provides a set of criteria and standards which must be achieved to meet the expectations of the customer
- Outlines a plan of activities which will assure the customer that the quality targets will be met (i.e. a Quality Assurance Plan)
- Identifies the techniques used to control the actual level of quality of each deliverable as it is built (i.e. a Quality Control Plan).

Finally, it is important to review the quality not only of the deliverables produced by the project but also of the management processes which produce them. A summary of each of the management processes undertaken during the execution phase is identified, including Time, Cost, Quality, Change, Risk, Issue, Procurement, Acceptance and Communications Management.

5. Develop Risk Plan

The foreseeable project risks are then documented within a Risk Plan and a set of actions to be taken formulated to both prevent each risk from occurring and reduce the impact of the risk should it eventuate. Developing a clear Risk Plan is an important activity within the planning phase as it is necessary to mitigate all critical project risks prior to entering the Execution phase of the project.

6. Develop Acceptance Plan

The key to a successful project is gaining acceptance from the customer that each deliverable produced meets (or exceeds) his/her requirements. To clarify the criteria used to judge each deliverable for customer acceptance, an Acceptance Plan is produced. The Acceptance Plan provides the criteria for obtaining customer acceptance, a schedule of acceptance reviews within which customer acceptance will be sought and a summary of the process used to gain acceptance of each deliverable from the customer.

7. Develop Communications Plan

Prior to the Execution phase, it is also necessary to identify how each of the stakeholders will be kept informed of the progress of the project. The Communications Plan identifies the types of information to be distributed, the methods of distributing information to stakeholders, the frequency of distribution and responsibilities of each person in the project team for distributing information regularly to stakeholders.

8. Develop Procurement Plan

The last planning activity within the Planning phase is to identify the elements of the Project which will be acquired from external suppliers to the project. The Procurement Plan provides a detailed description of the Products (i.e. goods and services) to be procured from suppliers, the justification for procuring each

product externally, as opposed to from within the business, and the schedule for procurement. It also references the process for the selection of a preferred supplier ("Tender Process") and the process for the actual order and delivery of the procured products ("Procurement Process").

4.7 The Software Development Plan (SDP)

The project development plan is one of the first formal documents produced by the project. Within this document, the project manager describes in detail:

- *How the project will be developed?*
- *What resources will be required?*
- *How these resources will be used?*

The project development plan assures that the development of the project is well charted before the main development activities begin. In -addition to the basic development schedule, the plan addresses such issues as:

- *The timely provision of equipment and tools so that they are available to developers when needed.*
- *The availability of staff to perform the development tasks in accordance with the schedule.*
- *Provision of contingency plans in the event that project risks materialize*
- The designation .of duties within the development team, and the assignment of these duties to the team members.

Preparing the project plan for a software project helps you ensure that the specified requirements and objectives are met successfully. It is a collation of all planning activities that have happened for a software project. This includes activities such as design and analysis, activity definition, risk planning, and cost estimation. To create the plan, you assess all planning activities, organizational policies regarding the creation of the project plan and assumption and constraints for the project. To implement the software project plan, you require management skills, such as leadership, communication, and problem solving, along with the basic knowledge about the software. You also need to ensure that the senior management bf the company has authorized work on the software project. Knowledge management techniques help you to make informed decision regarding the project plan.

After the project plan is executed, you manage the changes to it in such a way that the performance measurement baselines are not impacted, To manage the project plan effectively you monitor the project plan, periodic performance status reports, and requests for change. The primary tool that you can use to control the changes in the project plan is the change control mechanism: This is a set of formal procedures for changing the project plan.

4.7.1 Software Development Plan (SDP) Information needs

The Project Plan is a vital part of the project initiation stage. The plan should normally contain the following information:

1. Introduction and status of the plan
2. The authorisation procedures
3. Statement of project objectives
4. Statement of requirement
5. Deliverables in the project
6. A Work Breakdown Structure
7. The project milestones
8. The resource requirements
9. Interdependencies of work
10. The timetable of events
11. Staffing, organisation and responsibilities
12. Development methods and toolsets to be used
13. Source documentation
14. Resource and financial summary

This information creates the generation of a Project Book (log). The log should be in a loose-leaf binder with clearly identified sections and version control exercised over the documentation sets. These logs are now often retained as computer file, which enables a greater level of security to be maintained over them and version control to be established as an automatic feature.

4.7.2 Software Development Plan (SDP) Steps/items required

The contents of the project development plan may be adapted to the size of the project; it may be a large document or just a few pages. Table 1 presents an outline of some of the subjects covered in the project development plan.

Not all subjects in Table 1 are applicable to all projects. For example, many projects do not administer their own budget. Some organizations have a financial officer responsible for the administration of project budgets. The interface with external sources is another area riot applicable to all projects. The term, external sources covers such activities as interfacing with subcontractors, vendors and representatives of the customer.

Many standards have been produced for the project development plan. The formal structure or the project development plan document differs, depending on the actual documentation standard used. For example, the US DOD standard 2167 provides the option of describing the testing, configuration management and quality assurance plans in three separate documents. For large projects, this option can become a requirement.

The IEEE standard 1058.1 describes what is referred to as the software project management plan, which is essentially the same as the project development plan. This standard is largely compatible with the 2167 project development plan, although it is significantly less detailed.

This standard, too, provides the option of including configuration management and quality assurance plans, or of describing them in separate documents. The project development plan should be prepared as a standalone document, in the sense that it should be read and understood without the need to refer to other documents.

A general overview of the project is therefore usually included in the first section of the document. References for additional detail, of course, should always be provided, including pointers to such documents as the project contract, the concept document or the market research analysis.

Table 1: Software project development plan items

1. **System overview**
2. **Software development management**
   - Project organization and resources
   - Development facilities
   - Project organizational structure
   - Personnel
3. **Schedule and milestones**
   - Scheduled activities
   - Milestones and baselines
   - Activity network diagrams
   - System component source
   - Budget administration
   - Milestone payments
   - Major budgetary expenditures
   - Expenditure authorization procedure
4. **Risk analysis**
5. **Security**
6. **Interface with external sources**
7. **Procedure for formal reviews**
8. **Corrective action process**
9. **Problem change report**
10. **Software engineering**
    - Standards and procedures
    - Development methodology
    - Development resources
    - Personnel - qualifications and function
11. **Testing procedure**
12. **Software configuration management**
13 **Software quality assurance**

4.7.3 Inputs to SDP

Project plan development uses the outputs of the other planning processes, including strategic planning, to create a consistent, coherent document that can be used to guide both project execution and project control. This process is almost always iterated several times. For example, the initial draft may include generic resource requirements and an undated sequence of activities while the subsequent versions of the plan will include specific resources and explicit dates.

1. **Other planning outputs.** All of the outputs of the planning processes in the other knowledge areas are inputs to developing the project plan. Other planning outputs include both base documents, such as the WBS, and the

supporting detail. Many projects will also require application area-specific inputs (e.g., most major projects will require a cash-flow forecast).

2. **Historical information.** The available historical information (e.g., estimating data-bases, records of past project performance) should have been consulted during the other project planning processes. This information should also be available during project plan development to assist with verifying assumptions and assessing alternatives that are identified as part of this process.

3. **Organizational policies.** Any and all of the organizations involved in the project may have formal and informal policies whose effects must be considered. Organizational policies that typically must be considered include, but are not limited to:

    - ***Quality management***—process audits, continuous improvement targets
    - ***Personnel administration***—hiring and firing guidelines, employee performance reviews
    - ***Financial controls***—time reporting, required expenditure and disbursement reviews, accounting codes, and standard contract provisions.

4. **Constraints.** A constraint is an applicable restriction that will affect the performance of the project. For example, a predefined budget is a constraint that is highly likely to limit the team's options regarding scope, staffing, and schedule. When a project is performed under contract, contractual provisions will generally be constraints.

5. **Assumptions.** Assumptions are factors that, for planning purposes, are considered to be true, real, or certain. Assumptions affect all aspects of project planning, and are part of the progressive elaboration of the project. Project teams frequently identify, document, and validate assumptions as part of their planning process. For example, if the date that a key person will become available is uncertain, the team may assume a specific start date. Assumptions generally involve a degree of risk.

LECTURE # 24

4. PLANNING

4.7.4 Tools and Techniques for SDP

6. **Project planning methodology**. A project planning methodology is any structured approach used to guide the project team during development of the project plan. It may be as simple as standard forms and templates (whether paper or electronic, formal or informal) or as complex as a series of required simulations (e.g., Monte Carlo analysis of schedule risk). Most project planning methodologies make use of a combination of "hard" tools, such as project management software, and "soft" tools, such as facilitated startup meetings.

7. **Stakeholder skills and knowledge.** Every stakeholder has skills and knowledge that may be useful in developing the project plan. The project management team must create an environment in which the stakeholders can contribute appropriately. Who contributes, what they con-tribute, and when they contribute will vary. For example:

    - On a construction project being done under a lump-sum contract, the professional cost engineer will make a major contribution to the profitability objective during proposal preparation when the contract amount is being determined.
    - On a project where staffing is defined in advance, the individual contributors may contribute significantly to meeting cost and schedule objectives by reviewing duration and effort estimates for reasonableness.

8. **Project management information system (PMIS).** A PMIS consists of the tools and techniques used to gather, integrate, and disseminate the outputs of project management processes. It is used to support all aspects of the project from initiating through closing, and can include both manual and automated systems.
9. **Earned value management (EVM).** A technique used to integrate the project's scope, schedule, and resources and to measure and report project performance from initiation to closeout.

4.7.5 Outputs from SDP

1. **Project plan**: The project plan is a formal, approved document used to manage project execution. The project schedule lists planned dates for

performing activities and meeting milestones identified in the project plan. The project plan and schedule should be distributed as defined in the communications management plan (e.g., management of the performing organization may require broad coverage with little detail, while a contractor may require complete details on a single subject). In some application areas, the term *integrated project plan* is used to refer to this document. A clear distinction should be made between the project plan and the project performance measurement baselines. The project plan is a document or collection of documents that should be expected to change over time as more information becomes available about the project. The performance measurement baselines will usually change only intermittently and then generally only in response to an approved scope of work or deliverable change.

4.7.6 SDP Execution

Project plan execution is the primary process for carrying out the project plan the vast majority of the project's budget will be expended in performing this process. In this process, the project manager and the project management team must coordinate and direct the various technical and organizational interfaces that exist in the project. It is the project process that is most directly affected by the project application area in that the product of the project is actually created here. Performance against the project baseline must be continuously monitored so that corrective actions can be taken based on actual performance against the project plan. Periodic forecasts of the final cost and schedule results will be made to support the analysis.

4.7.7 Inputs to SDP Execution

1. **Project plan**: The subsidiary management plans (scope management plan, risk management plan, procurement management plan, configuration management plan, etc.) and the performance measurement baselines are key inputs to project plan execution.
2. **Supporting detail**
3. **Organizational policies**: Any and all of the organizations involved in the project may have formal and informal policies that may affect project plan execution.
4. **Preventive action**: Preventive action is anything that reduces the probability of potential consequences of project risk events.
5. **Corrective action**: Corrective action is anything done to bring expected future project performance in line with the project plan. Corrective action is an output of the various control processes—as an input here it completes the feedback loop needed to ensure effective project management.

4.7.8 Tools and Techniques for SDP Execution

1. **General management skills:** General management skills such as leadership, communicating, and negotiating are essential to effective project plan execution.
2. **Product skills and knowledge:** The project team must have access to an appropriate set of skills and knowledge about the project's product.
3. **Work authorization system.** A work authorization system is a formal procedure for sanctioning project work to ensure that work is done at the right time and in the proper sequence. The primary mechanism is typically a written authorization to begin work on a specific activity or work package. The design of a work authorization system should balance the value of the control provided with the cost of that control. For example, on many smaller projects, verbal authorizations will be adequate.
4. **Status review meetings**. Status review meetings are regularly scheduled meetings held to exchange information about the project. On most projects, status review meetings will be held at various frequencies and on different levels (e.g., the project management team may meet weekly by itself and monthly with the customer).
5. **Project management information system**.
6. **Organizational procedures**. Any and all of the organizations involved in the project may have formal and informal procedures that are useful during project execution.

4.7.9 Outputs from SDP Execution

1. **Work results**. Work results are the outcomes of the activities performed to accomplish the project. Information on work results—which deliverables have been completed and which have not, to what extent quality standards are being met, what costs have been incurred or committed, etc.—is collected as part of project plan execution and fed into the performance reporting process. It should be noted that although outcomes are frequently tangible deliverables such as buildings, roads, etc., they are also often intangibles such as people trained who can effectively apply that training.

2. **Change requests**. Change requests (e.g., to expand or contract project scope, to modify cost [budgets], or schedule estimates [dates, etc.]) are often identified while the work of the project is being done.

LECTURE # 25

4. PLANNING

4.7.4 Elements of SDP

Project Plan is iteratively defined through Concept & Requirements Phase. Initial estimates are refined as scope and requirements become clearer. There are two phases of project plan:

1. Preliminary
2. Final

Following are some elements of Software Project Plan:

a) Scope Planning

- Scope planning is the process of progressively elaborating and documenting the project work (project scope) that produces the product of the project.
- Project Scope Planning starts with the initial inputs of product description, the project charter, and the initial definition of constraints and assumptions.
- **Note that the product description incorporates product requirements that reflect agreed-upon customer needs and the product design that meets the product requirements.**
- The outputs of scope planning are the scope statement and scope management plan, with the supporting detail.
- The scope statement forms the basis for an agreement between the project and the project customer by identifying both the project objectives and the project deliverables.
- Project teams develop multiple scope statements that are appropriate for the level of project work decomposition.

b) Objectives – Business Requirements

- Could be "Mission Statement"
- Defines Business objectives for project
- Includes Business Case (NPV Model) & Detailed Description of assumptions.

c) Technical Approach

- Description of how new system is to be developed
  - Technologies
  - In House vs. Consultants
  - Derivatives of existing (i.e. use existing object model)
  - Architectural Layout – Layers

d) Contractual Aspects

- Specifies general needs from outside sources
  - Consultants
  - Software Suppliers
  - Hardware Suppliers
  - Network/Infrastructure Suppliers

- There are two types of contracts
  1. **Cost -Plus**
  2. **Fixed Price**

Most other relationships are some kind of combination of these two

1. Cost-plus (also called Time and material)

Cost-plus is a contractual relationship where the developer is paid for the cost of the service provided and in addition is allowed an agreed profit margin.

This is rather like renting a car the customer pays for the time that the car is used (by the hour, day, week etc.), and for any other expenses such as insurance and gasoline.

2. Fixed price

A fixed price contract is a commitment by the developer to provide an agreed product or service for an agreed fee, within an agreed schedule.

This is similar to purchasing a bus ticket, when the bus company agrees to take the customer to a specific destination within a published timetable, and for an agreed fee.

e) Schedules

- Defines specific dates for milestones components
- Work Breakdown Structure
- Use Scheduling Engines like MS project etc.

f) Resource Allocation

- Resources such as licenses, servers, or other software/hardware related items
- Personnel – how many people of what type and for how long

(Remember these are initial estimates in preliminary project plan)

g) Evaluation Methods

- What methods to be used to validate performance
  - Testing for adherence to spec
  - Monitoring usage
    - Web trends
    - Database Logs
  - Transaction logs
  - Setup schedule for reviews

h) Overview of Project Management

- Define where possible problems can occur
  - New technologies
  - Business risks
  - Resources
- Define Contingency Plans
  - Development methods
  - If the worst happens, what to do

LECTURE # 26

5. ORGANIZATION

4.8.1 Life cycle Models

To ensure the successful execution of a project, it is necessary to break down the project into: multiple manageable tasks.

Each task is performed in a series using processes. To understand what a process is, consider an example of a non software-related project. You are planning the market launch of an office management product. To create an effective plan, you need to perform certain tasks.

- First, you schedule a meeting of all managers and the Finance, Marketing, Production, and Systems personnel. You can follow a process to complete this 'task. You may send email messages or call them up personally.

- Next, you decide some feasible marketing and advertising strategies. Again there are processes that help you select the strategies.

- Fina1ly, you determine the territory where the product should be launched. This is done in consultation with the Production and Marketing departments.

Just like a non software-related project plan consists of multiple processes, a software development activity also consists of multiple tasks. A process or a combination of multiple processes is required to complete each task. A typical SDLC follows a consistent sequence of processes or a process model.

A process is defined as a collection of related tasks with specific milestones. To ensure smooth progress of a software project, relevant processes are arranged and executed in a sequence. Every software project is discrete with respect to its complexity, size, and goals. Therefore, different process models are designed for different software projects. These process models provide approaches that decide the path of software development from the conceptualization of a project to its formal termination.

1. **The *Waterfall* model**: This is the traditional life cycle model. It assumes that all phases in a software project are carried out sequentially and that each phase is completed before the next is taken up.
2. **The *Prototyping* Model**: A model that works on an iterative cycle of gathering customer requirements, producing a prototype based on the requirement specifications, and getting the prototype validated by the

customer. Each iteration of the life cycle builds on the prototype produced in the previous iteration.

3. **The *Incremental* Model**: The Incremental model is an example of an evolutionary life cycle model. It combines the linear nature of the Waterfall model and the iterative nature of the Prototyping model. The Incremental model divided the development life cycle into multiple linear sequences, each of which produces an increment of the final software product. In this model, the software product is developed in builds. A build is defined as a self-contained unit of the development activity. The entire development cycle is planned for a specific number of logical builds, each having a specific set of features.
4. **The *Spiral* model**: Another evolutionary life cycle model that combines the linear nature of the Waterfall model and the iterative nature of the Prototyping model. The project life cycle is divided into phases, and each phase is executed in all of the iteration of the Spiral Model.
5. **The RAD Model**:

A Process defines the overall processes that an organization needs to follow to manage a project efficiently. It defines a structured approach to sequence the phases and identify the requirements of each phase in the SDLC. The definition of the phases in a process model helps organize, monitor, and execute a project efficiently.

A process model is flexible and does not follow any rigid rules of implementation. An organization can deploy a totally new process model for its overall software development effort. It can also customize an existing process model to merge with its implemented process model. For example, figure 1, shows two process models, A and B. Process model A was used by an organization that manufactures confidential defense applications. The management used the model to plan extensively for unexpected project risks. However, after discussing with the project managers and system analysts, the organization wants to merge its current process model with a model lat has simple and consistent SDLC phases. This leads to the evolution of the process model B.

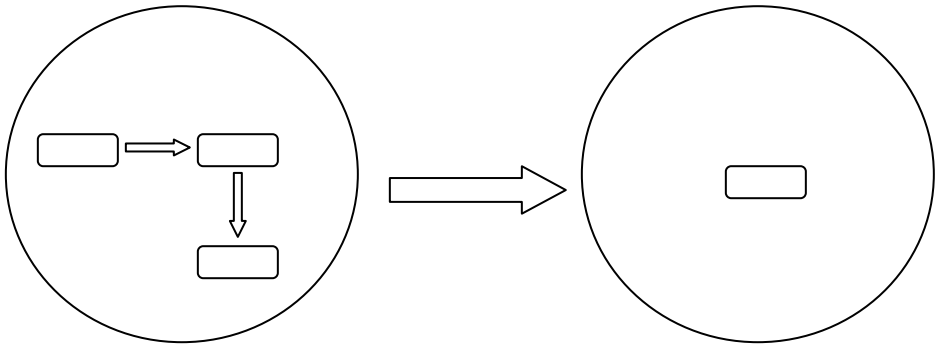


Figure: Customizing Process Models

4.8.2 Choosing the lifecycle model

In choosing a lifecycle model for your project, you should examine:

1. How well requirements are understood at the beginning? Is it going to change when moving through the project?
2. Is system architecture understood? Any changes on the way?
3. Level of reliability?
4. Level of re-design for future versions?
5. Risk Level?
6. Stuck With Predefined Schedule?
7. Need For Midcourse Correction?
8. Customer Informed through the Project?
9. Visible Management through the Project?
10. If a Model is chosen, how much it needs Modification?

Selecting an appropriate process model is crucial because it can provide a basic framework to initiate and carry out a project to its conclusion. It also defines the path for various project- related activities.

For example, if you select a process model, you know for sure that you need to carry out certain activities, such as planning, scheduling, resource allocation, risk management, and cost and effort estimation. These activities ensure smooth progress of a project within the

allocated time and ensure maintenance of quality and measurability throughout the project.

With so many process models available, a project manager is likely to face the dilemma of selecting the ***right*** process model for managing a software project efficiently.

To select a process model that is suitable to a project, the following criteria can be considered:

- Business goals of the organization
- Expected size of the project
- Client and project requirements
- Availability of funds and development staff
- Risks perceived in the project

Business Goals of the Organization
This criterion indicates the overall approach and mindset of an organization. If the organization has a past history of developiI;1g projects in accordance with well-defined plans and other job aids in every phase, the suitable process model can be the Waterfall model. However, if the organization is well equipped with the resources to deal with financial, technological, and personnel risks, it can choose the Spiral model. The organization can choose the Prototyping model if it is used to working in an experimental and a constant feedback mode.

Expected Size of the Project
If the size of a project is extensive and the client prefers all the features of the proposed project at the first delivery, you can select the Waterfall model. When you want the entire product to be developed and delivered in piecemeal so that the clieI;1t can immediately begin the unit testing of each module, you would select the Incremental model. In contrast, if the size of the project is doubtful, you can go for the Prototyping or Spiral model. These models help to develop projects that have a vague and uncertain estimate of the project size.

Client and Project Requirements
The Waterfall model or the Incremental model is chosen if the client needs and the project requirements are defined and approved. In addition, no changes or negligible changes are 1 expected in the future regarding design requirements. In contrast, you would choose the Spiral j and Prototyping models when the client needs and system requirements are uncertain and is likely to change in the future.

Availability of Funds and Development Staff
The Waterfall and Prototyping models require predetermined and adequate resources at the start of a project. However, if you expect additional funds and human resources as you progress through the different phases, you should go in for the Incremental or Spiral models. This is because the Incremental model operates on the assumption of developing a project into several builds due to lack of human resources. Similarly, the funds and

staffing requirements in the Spiral model may increase or decrease depending on the changes in requirements arid feasibility of the proposed project in the future.

Risks Perceived in a Project

This is yet another important criterion for the selection of a process model. You choose the Waterfall or Incremental model if the occurrence of risks and their impact perceived is minimum. However, you should go in for the Spiral model if the risks and their perceived impact are very high.

You should use the waterfall model if:

- The project is large yet it is clearly divided into discrete phases and each phase has defined number of days, personnel, and resources allocated. The Waterfall model requires the presence of defined phases and processes in each phase.

- The development also perceives that it can have defined set of input and certified output. This is because each phase would lead to another and the next phase would not begin until the previous phase is certified as closed. This way baselines and milestones for each phase can be identified. The Waterfall model assumes the closure of a previous phase before the successive phase begins. This ensures linear progression of a project where developers do not need to revert to an earlier phase or a process.

- The development team is not likely to face any bumps in the development process because of clear cut project requirements as well documentation provided to them by the client. Clarity of project vision and project requirements are the essential features of the Waterfall model, which are fulfilled in the preceding scenario.

Therefore, if the requirements are defined and a project is large enough to be divided into defined phases, you can go for the Waterfall model.

However Waterfall model has the following drawbacks:

- *There cannot be a sudden crossover from one phase to the next.* Real-time projects require sudden crossover between phases because such projects are subjected to change in every phase of the development cycle. For example, real-time projects such as embedded software development where the phases and the requirements for every phase cannot be determined at the analysis phase cannot deploy the Waterfall model.

- Another reason for the unpopularity of the Waterfall model is that it *requires too much effort for stringent documentation-in every phase.* Every phase is closed with formal documentation. Projects with predictable final product, such as banking software or an airline reservation project, usually have formal documentation. This is because the phases of these projects are defined. However, it is difficult for research and development (R&D) related projects to complete all project documentation.

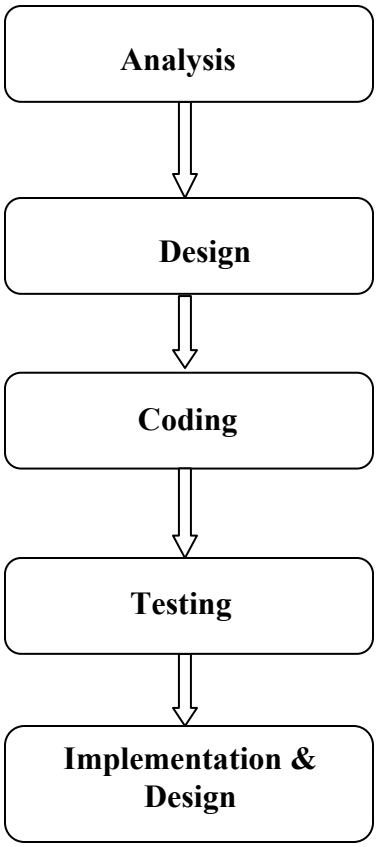
- The Waterfall model does *not support the development of a working model of a*

project first and then further development based on client feedback. Absence of a working model prevents you from detecting an issue in an early phase. As a result, you incur higher expenditure in rectifying the issue in a later phase. This in turn has an adverse effect on the effort, cost, and time spent on rework.

- *Finally, the Waterfall model causes, as a "blocking state".* When a blocking state occurs, some team members wait for other team members to finish a dependent task.

  For example, in the following figure, the team member assigned to do the design task cannot begin work until the analysis is complete. This delays the turnaround of the software project. Many times, the blocking state wastes a lot of developers' time that could have been spent on productive project -related work.

Figure: *Idle Team Members in the Phases of the Waterfall Model*



You should use Prototyping model when:

- The client is not clear about the requirements of the proposed system or

- When there is a conflict in client requirements.

To resolve the conflict, the development team develops a working model so that the requirements of the client become defined. Defined client requirements enable the physical development of the actual product.

The Prototyping model enables an analysis team to first construct a working model with their prior software experience combined with the vague needs of the client.

The Prototyping model can be as simple as a drawing on a paper or as complex as the real working software. The closer your prototype is to the actual product, the more precise is your evaluation.

There are different types of prototyping methods that an organization can implement:

- Rapid prototyping
- Reusable prototyping
- Modular prototyping

Rapid prototyping
This model is suitable when:

- The cost and time required to create a prototype is minimal.
- The project has a substantially long cycle
- Development team wants the design to be strong.

Reusable prototyping
This model is used when:

- The old design needs major changes but the supplementary components of the old design do not need major changes.

Modular prototyping
This model is used when:

- Considerable cost, time and effort are deployed to create a prototype.
- Client feedback for enhancements is not major
- Minor improvisations are needed to obtain client approval.

Using the Prototyping model saves cost and time involved in the build-it-twice approach. The experience of developing the prototype is useful for developers while developing the final product. It reduces the risks of an unfeasible project design because the developers gain a fair idea of the resources and the probable time taken to create the [mal product. They also get a feel of the implementation tool to be used in the project.

Incremental Model

You can use the Incremental model approach in software projects:

- Where human resource is scarce.

In such cases, the management can decide to divide and develop a product in individual builds. When the resources are available, the subsequent builds are planned and executed. This process model is useful in real life because normally all the resources required to complete the project are not available at the same time.

***Example**:*

Consider a situation in which you might need to use the Incremental model of SDLC. *Supersoft2000* requires a software product that automates the employee and their salary details. It has assigned this project to Blue Valley Consulting (BVC) that specializes in developing Human Resources (HR)-related applications. The project requirements are defined to start the project. However, the client wants to roll out the new system for the benefit of its employees as early as possible. The analysis team in BVC has been able to divide the entire project into seven independent modules. On the basis of their prior experience, it feels each module can be executed and deployed independently by the client. After all the modules are completed, the maintenance team in BVC can assemble and implement the system at the client site. Currently, the development personnel in BVC are hired on contract and BVC faces shortage of skilled personnel.

In such a case, you use the Incremental model because of the following reasons:

- Time and lack of skilled personnel are the main hindrances in this project.

Therefore, by using the Incremental model, you can design, develop, and deliver independent modules. This way by using a few skilled personnel you are able to develop a system with basic features and provide it to the client immediately. The most important and the least dependent module are developed first. While the client is using the module, BVC can arrange for additional skilled personnel to complete the rest of the modules and deliver them to the client.

The Incremental model enables you to revert to an earlier phase and refine the product based on client feedback. After all the modules have been developed, the development personnel can be released. BVC can then hire or retain a few experienced personnel to create a maintenance team. This team would assemble and implement the entire employee salary details product at the client site. This flexibility of personnel can be exercised because you use the Incremental model. Using the Incremental model enables:

- Division of skilled labor
- Division of a complex project into modules
- Development of those modules within a short time frame

Therefore, if *time and skilled personnel* are constraints of a Project, you can effectively use the Incremental model.

Spiral Model
The basic goal of using the Spiral model is to define ways of eliminating risks in the design phase. Consequently, minimum and manageable risks percolate into the development phase.

Example:

LMN Inc. has acquired a project to develop a telecommunications project using the Voice-over Internet telephony (VOIP) technology. The company does not have any prior experience or the required level of expertise to develop such a project. There is a team of three analysts and the company does not propose to dedicate a team to complete this project. This is because it anticipates many risks and a large amount of rework that may add to the cost of project execution. The company has determined multiple models and designs to execute the project. After presenting the models to the client, the client is itself confused. However, it assures LMN Inc. of continued feedback and support. The client fully understands the ambitiousness of the project and does not consider time and budget as constraints for the project. Consequently, LMN Inc. decides to use the client feedback and risk analysis effectively until the end of the project.

In this situation, you would use the *Spiral model* because of the following reasons:

- LMN Inc. has **no prior experience or expertise** in executing such a project. Therefore, it expects a lot of rework in reverting to an earlier phase and incorporating client feedback therein. Using the Spiral model, you can to revert to an earlier phase to incorporate client feedback and compete that phase.
- It needs to **perform risk analysis effectively** to eliminate losses arising due to uncertain development models, resource requirements, project constraints, and time.
- It is also preferable to create a **prototype for every deliverable** that LMN Inc. might deem necessary to receive client feedback.
- **Spiraling cumulative project costs** can be covered up by the **liberal budget** provided by the client.

4.9 Planning Documents

You (the PM) need to choose which documents are appropriate. Documents do not have to be lengthy.

The requirements phase produces one main product document:

- ***The software requirements specification document***

  And two Project Planning Documents:

1. ***The Project Development Plan***
- ***The Software Test Plan***

The requirements phase formally concludes with the project's first major review: the software requirements review (SRR). It is this review that signs off the requirements specification and formally declares the requirements document as the first approved project baseline.

2. ***The Statement of Requirement (SOR)***

   The project manager (or the advising consultant adopting this role) must ensure that the project sponsor has produced a written statement of requirements (SOR). This must be a thorough document which is:

   - *Unambiguous*
   - *Fully defined or complete*
   - *Verifiable deliverables*
   - *No conflicts*
   - *Consistent*
   - *Auditable*

   The SOR will be the document against which change control will be exercised. The SOR should be closely matched to the contract and there should be no conflict of interests between the two. Where consultants are involved, the client or sponsor SOR will normally form the basis of the proposal. All of these documents must carefully align and there should be no scope for misinterpretation, confusion or lack of understanding. This will be the cornerstone of the project.

3. ***System Specification***

   If a *System Specification* has been properly developed, nearly all information required for a description of software scope is available and documented before software project planning begins. In cases where a specification has not been developed, the planner must take on the role of system analyst to determine attributes and bounds that will influence estimation tasks.

4. ***Design Specification***

   The first stage of risk analysis requires a review of all, project technical and administrative plans in order to identity potential problems. It includes:

   - *The project development plan*
   - *The requirements specification*
   - *The design specification*

All major dependencies in the project development plan are examined and evaluated.

Examples may be the dependence on external sources such as subcontractors, vendors and suppliers, and service providers. Problems will arise if external components or services are not provided on time, or if they do not function as expected.

The project design specification is a detailed plan of how the requirements are to be implemented.

The implementation decisions involved may contain potential problems, For example, problems will arise if the selected hardware turns out to be inadequate, such as if the CPU is too slow, the LAN is not sufficiently reliable, or the maximum available memory is insufficient.

A list of all anticipated problems is then compiled, identifying each problem and describing the potential effect on the project. Following table presents an example of an anticipated problem list.

Table 1: Example of an anticipated problem list

Problem	*Description*
1. **Late delivery of the Development computer**	If the computer is not delivered by June 1, as planned the integration phase will be delayed.
2. **Insufficient memory**	The size of the memory resident part of the System may exceed 8 megabytes (the maximum memory size supported by the computer).
3. **No operating system**	The system requires changes to the standard expert operating system. John Adams is the only OS Expert in the company, and he may not be available for this project.
4. **System response time too slow.**	The required system response time to the input may exceed the 5 seconds specified in the requirements.
5. **High staff turnover**	The schedule is tight with only minimal slack time. If there is more than average staff Replacement during development, we will slip the schedule.

6.	**Communication too slow**	The standard communications package is too slow. The design is based on the new binary Communication package. This package has never Been used with this system and may not be suitable.
7.	**Late delivery of the Data base subsystem**	The data based subsystem is subcontracted to Software Developer Inc. (SDI), who have Committed to delivery by April 15, SDI may not Deliver on time, thus delaying the final integration and test phase.

The anticipated problem list should be compiled with the participation of the principle members of the project development team. Other people may also be invited to contribute to the list, based on their experience and technical or administrative knowledge. This might include people from other project teams, support groups, the company's legal department or the purchasing department.

While the objective is not to list every conceivable problem that any project may experience, it is necessary to identify those problems that should reasonably be considered in relation to the project. In any event, the following analysis stage is designed to isolate only those problems that could have significant impact on the project, and that can reasonably be expected to appear.

5. ***The Data Item Descriptions (DID s)***

Standard 2168 (DOD 1988b) contains the requirements for the development, documentation and implementation of a software quality program. The software quality program includes planning for and conducting:

- Evaluations of the quality of software
- Associated documentation and related activities
- Follow-up activities necessary to assure timely and effective resolution of problems

The DID s are a comprehensive set of documentation standards that cover all phases of software development, maintenance and the production of user reference manuals. The DID s include a section called *preparation instructions* that provide a large degree of freedom by permitting tailoring of the document format and the use of alternate presentation styles. The full set of DID s is described in Table 2.

Standard 2167 states that it is not intended to specify or discourage the use of any particular software development method (DOD 1988a). However, as mentioned previously, the standard is heavily inclined toward phased development methodologies, such as the Waterfall paradigm. The phased approach is inherent

in the required development stages and reviews, requiring system design to be followed by software requirements, software design, implementation and testing.

The Data Item Descriptions define the formal documentation standards for all required documents generated during the development of software according to standard 2167. DID s apply to the development of one or more computer system configuration items (CSCI s), a term used throughout the 2167 standard to identify high level decomposition components of a computer system.

Table 2: DOD Data Item Descriptions (DID s)

Development documentation
- Software development plan

System documentation
- System/segment specification
- System/segment design document

Software design and requirements
- Software requirements specification
- Software design document

Interface design and requirements
- Interface requirements, specification
- Interface design document

Version description
- Version description document

Test documentation
- Software test plan
- Software test description
- Test report

Release manuals
- Computer system operator's manual
- Software user's manual
- Software programmer's manual
- Firmware support manual

Maintenance documentation and source code
- Computer resources integrated support document
- Software product specification

A CSCI, as applied to software; is a component of the system that can be individually controlled, configured, tested and documented. CSCI s are often reviewed and approved as separate development items, and though a single review or audit can consider more than one CSCI, each is usually addressed separately during the review process. There are no clear-cut guidelines for the division of a software system into CSCI s as the division is essentially one of convenience.

Table 3: DOD Data Item Description standards

Data requirements title	Acronym
1. System segment design document	SSDD
2.Software development plan	SDP
3.Software requirements specification	SRS
4. Interface requirements specification	IRS
5. Interface design document	IDD
6. Software design document	SDD
7. Software product specification	SPS
8. Version description document	VDD
9. Software test plan	STP
10. Software test description	STD
II. Software test report	STR
12. Computer system operator's manual	CSOM
13. Software user's manual	SUM
14. Software programmer's manual	SPM
15. Software support manual	FSM
16. Computer resources integrated support document	CRISD
17. Engineering change proposal	ECP
18. Specification change notice	SCN

Table 3 contains a list of the DID s referenced by standard 2167A. The software quality DID is referenced separately in the DOD software quality program standard 2168. All DID document formats follow a similar pattern. Several of the sections are common to most, if not all of the documents, such as:

- *Title page format*
- *Table of contents*
- *Scope (including identification, overview, references etc.)*
- *Other applicable documents*
- *Notes and appendices*

phasis is on the required content and not on the required format. This is specifically addressed in the preparation instructions accompanying each DIn addition, the page format, page numbering scheme, section numbering scheme and various other preparation instructions are common. This clearly suggests the use of an automatic tool to assist in the preparation of the documents, a practice greatly encouraged by the 2167 standard. Many such tools have been developed to support 2167, and Polack, in a paper analyzing the use of CASE tools for DOD projects (Polack 1990), concludes that these tools do indeed save time, and result in a higher quality software system.

Each DID describe the requirements for the preparation of a specific document, but the main emID, which state that other presentation styles, including charts, tables or matrices, are acceptable (e.g. Hatley and Pirbhai (1988) or Ward and Mellor (1986). There is also substantial flexibility in the requirements regarding the content of the documents. The standard provides for considerable tailoring to adapt the standard to the type of project being developed.

Tailoring the standard

Tailoring of standard 2167 is not only encouraged, it is required. The foreword to 2167 states that the standard must be appropriately tailored by the Program Manager to ensure that only cost-effective requirements are cited in defense solicitations and contracts.

The DOD has issued a guide for tailoring that can be used as a reference source for adapting the standards to the type of project being developed. Two basic principles apply to tailoring:

- The tailoring process is the *deletion* of non-applicable requirements.
- Tailoring of the standard should be carried out by the contracting agency.

The first principle means that these modifications can only include the deletion of requirements from the standard (and not changes to the requirements in the standard). The second principle means that the contractor (i.e. the developer) cannot tailor the standard without receiving permission from the contracting agency (i.e. the DOD).

Tailoring of the 2167 standard must be completed as early as possible. This is best performed either during contract negotiations or as one of the initial activities as soon as the project begins. The following is a description of the basic procedure for tailoring standard 2167:

1. Review all standard 2167 requirements, including;

    - reviews and audits
    - documentation

- testing activities
- quality assurance activities
- configuration control activities
- other required development activities

2. Identify the requirements that are not applicable, justifiable or reasonable for the project being developed. For example, the Firmware Support Manual will not be required if no firmware is being developed: or two design reviews (POR and COR) may not be necessary for a small project.

3. Prepare a list of requests for deletions from the standard. This may include:

   - exclusion of documents
   - exclusion of sections in documents
   - exclusion of activities.
   - exclusion of parts of activities

4. Prepare a written description of the justifications for each item that is requested to be tailored out.

5. Submit the tailoring request, together with the justification, as early as possible (preferably before the project begins).

In order to be able to differentiate between forgotten items and tailored items, all tailored items must be clearly referenced. When submitting a list of documents for a formal review or milestone, all documents tailored out should be listed together with a statement to that effect. Within a document, when a paragraph is tailored out a statement to that effect will appear directly after the paragraph number. If a paragraph and all of its subparagraphs are tailored out, only the highest level paragraph number need be included.

The list of the DID s together with the list of tailoring approvals are an integral part of the project deliverables. Until tailoring approval has been granted, the developer is obligated to provide the full list of Dills, with all their inclusions. This is the reason why tailoring should be concluded as early as possible.

6. ***The software configuration management plan (SCMP)***

The software configuration management plan (SCMP) is part of the project's software development plan. The SCMP may appear as a separate document or as a section within the project development plan.

The SCMP documents the resources that are needed, how they are to be used, and which standards and procedures will be applied during the project.

The SCMP then becomes the mandate for the configuration control group during project development. The issuance of this plan is the responsibility of the project manager, though in large projects it may be delegated to the configuration Control manager.

Table 4 contains a list of the main subjects covered in the SCMP. When any of these subjects is covered elsewhere (e.g. in the software quality assurance plan), it can be omitted from the SCMP and replaced by a pointer to the document in which it is covered. Though most of the subjects in Table 1 are self-descriptive, the following are some guidelines:

Configuration status accounting describes the way in which status information flows:

- From the developers to the configuration management organization (audits and reviews)
- From the configuration management organization to project management (status reporting procedures)

Configuration identification describes the method for designating development items as SCCI s. This is part of the high level decomposition of the system into major development components.

The section on identification methods describes the way in which each component generated by the project is marked for unique identification. Security, restricted access and classification refer to the secure development of sensitive products (such as documents, software, patents, military classified information etc.). It is often convenient to assign many of these tasks to configuration control because of the need to be involved in the review and classification of documents and other related activities that are associated with security.

Subcontractors, vendors and suppliers mayor may not implement their own configuration management plan. It is the project manager's responsibility to assure that either subcontractors or external developers submit a CMP for review, or that the project's configuration manager assumes responsibility for their work.

The SCMP may also include diagrams and flow charts to describe procedures for submitting change requests, or for reporting problems.

7. *Statistical software process improvement (SSPI)*

As an organization becomes more comfortable with the collection and use or process metrics, the derivation of simple indicators gives way to a more rigorous approach called *statistical software process improvement* (SSPI).

In essence, SSPI uses software failure analysis to collect information about all errors and defects encountered as an application, system, or product is developed and used. Failure analysis works in the following manner:

1. All errors and defects are categorized by origin (e.g., flaw in specification, flaw in logic, non conformance to standards).
2. The cost to correct each error and defect is recorded.
3. The number of errors and defects in each category is counted and ranked in descending order.
4. The overall cost of errors and defects in each category is computed.
5. Resultant data are analyzed to uncover the categories that result in highest cost to the organization.
6. Plans are developed to modify the process with the intent of eliminating or reducing the frequency of the class of errors and defects that is most costly.

Table 4: Example of the contents of a software configuration management plan

1. Software configuration management organization and resources
- Organization structure
- Personnel skill level and qualifications
- Resources

2. Standards, procedures, policies and guidelines

3. Configuration identification
- Method for defining SCCI s Description of the SCCI s for this project

4. Identification methods (naming and marking of documents, software components, revisions. releases, etc.)

5. Submission of configuration items
- Approval/rejection procedure

6. Change control
- Change control procedures (method of submission, review. approval and rejection)
- Reporting documentation (change requests, problem reports)
- Change review procedures and review board

7. Version control
- Preparation of software and documentation versions
- Release approval procedure

8. Storage, handling and delivery of project media
- Storage requirements
- Backups

9. Configuration control of subcontractors, vendors and suppliers

10. Additional control

- Miscellaneous control procedures
- Project specific control (security etc)

11. Configuration status accounting

- Configuration audits and reviews
- Configuration structure reporting procedures

12. Configuration management major milestones

13. Tools, techniques and methodologies

---

8. ***Communications management plan***

A communications management plan is a document that provides:

- A collection and filing structure that details what methods will be used to gather and store various types of information. Procedures should also cover collecting and disseminating updates and corrections to previously distributed material.

- A distribution structure that, details to whom information (status reports, data, schedule, technical documentation, etc.) will flow, and what methods (written reports, meetings, etc.) will be used to distribute various types of information. This structure must be compatible with the responsibilities and reporting relationships described by the project organization chart.

- A description of the information to be distributed, including format, content, level of detail, and conventions/definitions to be used.

- Production schedules showing when each type of communication will be produced.

- Methods for accessing information between scheduled communications.

- A method for updating and refining the communications management plan as the project progresses and develops. The communications management plan may be formal or informal, highly detailed or broadly framed, based on the needs of the project. It is a subsidiary component of the overall project plan.

9. ***The software quality assurance plan***

Every project must have a quality plan. The quality plan will be presented as a section in the project plan.

It is drawn up by the project manager at the start of the project and should be agreed with the project sponsor.

You would expect the quality plan to contain the following elements:

- Statement of the quality control organisation
- Identification of specific standards and methods that will be used
- Definition of the quality control procedures; this is aligned to the Work Breakdown Structure
- Specification of quality milestones
- Detail of unusual features
- Change control and configuration management
- Detail of acceptance procedures
- Specification of quality assurance procedures

The software quality assurance plan (SQAP), like the software configuration plan, is also part of the overall software project development plan.

The SQAP documents which resources are needed, how they should be used and which standards and procedures will be applied during the project.

The SQAP then becomes the mandate for the quality assurance group during project development. The issuance of this plan is the responsibility of the project manager, though in large projects it will usually be delegated to the quality assurance manager.

The SQAP may appear as a separate document or as a section within the project development plan, and may include the configuration management plan (if this has not been documented separately).

Table 5 contains a list of the main subjects covered in the SQAP. When any of these subjects is covered elsewhere, such as in the software configuration management plan (SCMP), it can be omitted from the SQAP and replaced by a pointer to the document in which it is covered.

However, the SCMP and the SQAP are concerned with different aspects of the controlled items. The SCMP is primarily concerned with the format of controlled items while the SQAP is more involved with the contents of controlled items.

The SQAP must cover subcontractors, vendors and suppliers, irrespective of whether or not these external entities have their own quality assurance organization.

In any project, the quality of external components is ultimately the concern of the project manager and the SQA organization. When a system fails, it usually makes

little difference whether the failure is due to an externally developed component or an in-house developed component.

The plans for supervising these external groups must be adapted to the type of external components being provided (off the shelf or new development) and the type of organization (do they have their own quality assurance organization?).

Table 5: Example of the contents of a software quality assurance plan

1. Software quality assurance organization and resources
- Organizational structure
- Personnel skill level and qualifications
- Resources

2. SQA standards. Procedures, policies and guidelines

3. SQA documentation requirements
- List of all documentation subject to quality control
- Description of method of evaluation and approval

4. SQA software requirements
- Evaluation and approval of software
- Description of method of evaluation
- Evaluation of the software development process
- Evaluation of reused software
- Evaluation of non-deliverable software

5. Evaluation of storage handling and delivery
- Project documents
- Software
- Data files

6. Reviews and audits

7. Software configuration management (when not addressed in a separate document)

8. Problem reporting and corrective action

9. Evaluation of test procedures

10. Tools techniques and methodologies

11. Quality control of subcontractors, vendors and suppliers

12. Additional control
- Miscellaneous control procedures
- Project specific control

13. SQA reporting, records and documentation.
- Status reporting procedures
- Maintenance
- Storage and security
- Retention period

The SQAP, as part of the project development plan, should be reviewed and updated periodically and whenever any requirements, project development procedures, methodologies or other relevant activities are changed.

The IEEE SQAP guide recommends periodic evaluation of two aspects of the plan:

(1) *The plan's content and*
(2) *The plan's implementation*

The plan's content should be evaluated with regard to the specific SQAP standard used, to assure the plan's continuing compliance with the standard even when the characteristics of the software project change.

The plan's implementation should be evaluated in terms of the changing scope of the project, including the tasks and responsibilities referenced in the plan, and other new or changed characteristics of the project.

When updating the SQAP the following project activities and events should be considered:

- New or changed contractual requirements
- Additional standards and policies
- Additional project documents
- Changes in the project's organizational structure
- New tools and utilities
- Additional subcontractors and vendors

10. *The RMMM Plan*

A collection of risk information sheets developed for all risks that lie above the cut off. A risk management strategy can be included in the software project plan or the risk management steps can be organized into a separate *Risk Mitigation, Monitoring and Management Plan.* The RMMM plan documents all work performed as part of risk analysis and are used by the project manager as part of the overall project plan.

Some software teams do not develop a formal RMMM document. Rather, each risk is documented individually using a risk information sheet (RIS) [WIL97]. In most cases, the RIS is maintained using a database system, so that creation and information entry, priority ordering, searches, and other analysis may be accomplished easily.

Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence.

As we have already discussed, risk mitigation is a problem avoidance activity. Risk monitoring is a project tracking activity with three primary objectives:

1. To assess whether predicted risks do, in fact, occur;
2. To ensure that risk aversion steps defined for the risk are being properly applied; and
3. To collect information that can be used for future risk analysis. In many cases, the problems that occur during a project can be traced to more than one risk. Another job of risk monitoring is to attempt to allocate origin (what risk(s) caused which problems throughout. the project).

11. ***Project Development Budget***

Good estimates are important, as they form the foundations of a good project development plan. This plan, prepared by the project manager, is produced during the initial stages of the project and includes estimates related to:

- The project development budget
- The project development schedule
- The required development resources (development staff, development equipment etc.)

In parallel with integration and testing, the following managerial and activities take place:

- Final budgeting of the project; the cost of changes is determined, risk contingency activities are evaluated, and the budget is updated.
- Training is conducted for users, operators, customers, installers, maintenance engineers, and marketing engineers.
- Installation sites are prepared, and the infrastructure for hardware and special equipment is planned and installed.
- The development team size is reduced.

12. ***Maintenance Documents***

The phased approach to software development divides the development life cycle into:

- The development of the software code
- Preparation for integration and test of the system (the next phase)
- The development of the maintenance plan

Apart from the actual code being written (and hopefully being well commented), some of the other documents that are developed during this phase include:

- The programmer's notebook, documenting coding decisions, unit tests, and resolution of implementation problems.
- Maintenance plan and documentation, including all necessary documentation needed for system maintenance.
- Initial versions of the user documentation, including reference manuals and operator guides.

At the conclusion of the integration and test phase all documentation must be complete and ready for delivery, including:

- *Maintenance documentation*
- *Final user documentation*
- *All updated development documentation*
- *Test documentation and test reports*

Maintenance requires a much smaller team, and a different type of management. In fact, a single maintenance group can be established to maintain several products, with common management, configuration control, installation and field engineers, and maintenance of documentation.

The documents that need to be updated during this phase include:

- *Version release documentation*
- *Problem reports*
- *All development documentation*
- *All user documentation*
- *Maintenance logs and customer service reports*

13. *The statement of work (SOW)*

The statement of work is the basis of the contract between the pro-poser and the customer, and is often incorporated into the contract. The SOW contains a detailed list of all work to be performed by the pro-poser for the benefit of the customer.

It is a narrative description of *products* or *services* to be supplied by the project. For internal projects, the project *initiator* or *sponsor* provides the statement of work based on business needs, or product or service *requirement*s. For external projects, the statement of work can be received from the *customer* as part of a bid document, for example, *request for proposa*l, *request for informatio*n, request for bid, or as part of a *contrac*t. The SOW indicates a:

- ***Business need*** - an organization’s business need, can be based on needed training, market demand, technological advance, legal requirement, or governmental standard.

- ***Product scope description*** - documents the product requirements and characteristics of the product or service that the project will be undertaken to create. The product requirements will generally have less detail during the initiation process and more detail during later processes, as the product characteristics are progressively elaborated. These requirements should also document the relationship among the products or services being created and the business need or other stimulus that causes the need. While the form and substance of the product requirements document will vary, it should always be detailed enough to support later project planning.

- ***Strategic plan*** - all projects support the organization's strategic goals—the strategic plan of the performing organization should be considered as a factor in project selection decisions.

The SOW starts as a general list of required deliverables in the RFP. A more detailed version t of the SOW is submitted as part of the proposal, and is still considered only an initial description of the work to be performed. The blinding version of the SOW is finalized during contract negotiations, or after the detailed project requirements have been completed.

Table 6 presents an example of an SOW outline for a software project. The list of items varies considerably, depending on the type of project being developed; for example not all projects include the delivery of hardware components, and not all projects require training or installation.

The basic guideline for the preparation of the SOW is that any activity, service or product required by the customer, and agreed to by the developer, must be included. This means that there can be no binding work items that were informally understood or agreed to verbally, which do not appear in the SOW.

The formal SOW must include all and only the work to be performed. This condition prevents misunderstandings and disagreements later, after the project begins.

The statement of work (SOW) describes the procurement item in sufficient detail to allow prospective sellers to determine if they are capable of providing the item. "Sufficient detail" may vary, based on the nature of the item, the needs of the buyer, or the expected contract form.

Some application areas recognize different types of SOW. For example, in some government jurisdictions, the term *SOW* is reserved for a procurement item that is a clearly specified product or service, and the term *Statement of Objectives* (SOO) is used for a procurement item that is presented as a problem to be solved.

The statement of work may be revised and refined as it moves through the procurement process. For example, a prospective seller may suggest a more

efficient approach or a less costly product than that originally specified. Each individual procurement item requires a separate statement of work. However, multiple products or services may be grouped as one procurement item with a single SOW.

Table 6*:* A sample SOW outline for a software project

1. **Referenced documents**
   - requirements specification
   - existing system description
   - customer's RFP
   - developer's proposal
   - vendor's and developer's technical literature

2. **Software deliverables**
   - functionality (as documented in the requirements specification)
   - list of major software components

3. **Equipment and hardware deliverables**
   - functionality (as documented in the requirements specification)
   - list of major hardware components

4. **Training**
   - user courses
   - operator training
   - installation training

5. **Market research**
6. **Procurement**
7. **Supervision of subcontractors**
8. **Documentation**
   - development documentation
   - user documentation
   - maintenance documentation
   - other technical documentation

9. **Testing**
   - alpha testing
   - beta testing
   - acceptance tests (ATP)

10. **Installation**
11. **Maintenance services**
12. **Other services and deliverable items**
13. **Method of delivery**

- software
- documentation
- hardware

The statement of work should be as clear, as complete, and as concise as possible. It should include a description of any collateral services required, such as performance reporting or post-project operational support for the procured item.

In some application areas, there are specific content and format requirements for a SOW.

14. ***Responsibility Assignment Matrix (RAM)***

Project roles (who do what) and responsibilities (who decide what) must be assigned to the appropriate project stakeholders.

Roles and responsibilities may vary over time. Most roles and responsibilities will be assigned to stakeholders who are actively involved in the work of the project, such as the project manager, other members of the project management team, and the individual contributors.

The roles and responsibilities of the project manager are generally critical on most projects, but vary significantly by application area. Project roles and responsibilities should be closely linked to the project scope definition.

A Responsibility Assignment Matrix (RAM) is often used for this purpose. On larger projects, RAM s may be developed at various levels.

For example, a high-level RAM may define which group or unit is responsible for each component of the work breakdown structure, while lower-level RAM s are used within the group to assign roles and responsibilities for specific activities to particular individuals.

A structure that relates the project organization structure to the *work breakdown structure,* to help ensure, that each element of the project's *scope* of work is assigned to a responsible individual.

– Shows who does what (x=person, y=phase). The most important feature of the RAM is the participatory development process involving all stakeholders. Show who is participant, who is accountable, who handles reviews, who provides input and who must sign off on specific work packages or project phases.

15. ***Project Charter***

A document issued by senior management that formally authorizes the existence of a *project*. And it provides the *project manager* with the authority to apply organizational resources to project activities.

16. ***Risk management plan***

The risk management plan describes how risk identification, qualitative and quantitative analysis, response planning, monitoring, and control will be structured and performed during the project life cycle. The risk management plan may include the following.

- **Methodology.** Defines the approaches, tools, and data sources that may be used to perform risk management on this project. Different types of assessments may be appropriate, depending upon the project stage, amount of information available, and flexibility remaining in risk management.

- **Roles and responsibilities.** Defines the lead, support, and risk management team membership for each type of action in the risk management plan. Risk management teams organized outside of the project office may be able to perform more independent, unbiased risk analyses of project than those from the sponsoring project team.

- **Budgeting.** Establishes a budget for risk management for the project.

- **Timing.** Defines how often the risk management process will be performed throughout the project life cycle. Results should be developed early enough to affect decisions. The decisions should be revisited periodically during project execution.

- **Scoring and interpretation.** The scoring and interpretation methods appropriate for the type and timing of the qualitative and quantitative risk analysis being performed. Methods and scoring must be determined in advance to ensure consistency.

- **Thresholds**. The threshold criteria for risks that will be acted upon, by whom, and in what manner. The project owner, customer, or sponsor may have a different risk threshold. The acceptable threshold forms the target against which the project team will measure the effectiveness of the risk response plan execution.

- **Reporting formats.** Describes the content and format of the risk response plan. Defines how the results of the risk management processes will be documented, analyzed, and communicated to the project team, internal and external stakeholders, sponsors, and others.

– **Tracking**. Documents how all facets of risk activities will be recorded for the benefit of the current project, future needs, and lessons learned. Documents if and how risk processes will be audited.

4.10 Scheduling

Any project can be completed, given an infinite amount of time and resources. Realistically, the amount of time available for project development is always finite.

In fact, in most cases it is less than the project manager considers sufficient.

Few projects are completed ahead of time; many projects overrun their schedule.

The project schedule is one of the most important parts of the project development plan.

The plan includes:

- Scheduling of development activities and
- Scheduling of project resources, particularly people

The project development plan describes in detail:

- how the project manager plans to develop the project
- what resources will be required and
- how these resources will be applied

No matter how well the project schedule is prepared, that schedule is useless unless it is adhered to. It is the project manager's responsibility to withstand pressure and to assure that the project is developed in an orderly fashion, according to the schedule. Whenever circumstances change, the project schedule should then be updated to reflect the new situation.

A schedule is a list of:

- Activities and
- Anticipated time of implementation of these activities

There are many ways of representing a schedule:

- Lists of activities,
- Diagrams,
- Graphs etc.

The most common methods of schedule representation are :

- precedence network diagrams (such as PERT),
- Gantt charts and
- lists of milestones

4.11 Guidelines for successful Planning

A common failure in many kinds of planning is that the plan is never really implemented. Instead, all focus is on writing a plan document.

Most of the following guidelines help to ensure that the planning process is carried out completely and is implemented completely or, deviations from the intended plan are recognized and managed accordingly.

i. Involve the Right People in the Planning Process

It's critical that all parts of the system continue to exchange feedback in order to function effectively. This is true no matter what type of system.

When planning, get input from everyone who will be responsible to carry out parts of the plan, along with representative from groups who will be affected by the plan.

Of course, people involved should be responsible to review and authorize the plan.

ii. Write Down the Planning Information and Communicate it Widely

New managers, in particular, often forget that others don't know what these managers know.

Even if managers do communicate their intentions and plans verbally, chances are that others won't completely hear or understand what the manager wants done.

Also, as plans change, it's extremely difficult to remember who is supposed to be doing what and according to which version of the plan.

Key stakeholders (employees, management, board members, sponsors, customers, clients, etc.) may request copies of various types of plans.

Therefore, it's critical to write plans down and communicate them widely.

iii. Goals and Objectives Should Be SMARTER

- Specific
- Measurable

- Acceptable
- Realistic
- Time frame
- Extending
- Rewarding

iv. Build in Accountability (Regularly Review Who's Doing What and By When?)

Plans should specify who is responsible for achieving each result, including goals and objectives.

Dates should be set for completion of each result, as well.

Responsible parties should regularly review status of the plan.

Be sure to have someone of authority "sign off" on the plan, including putting their signature on the plan to indicate they agree with and support its contents.

Include responsibilities in policies, procedures, job descriptions, performance review processes, etc.

v. Note Deviations from the Plan and Re-plan Accordingly

It's OK to deviate from the plan. The plan is not a set of rules. It's an overall guideline. It is equally important to notice deviations and adjust the plan accordingly.

vi. Evaluate Planning Process and the Plan

During the planning process, regularly collect feedback from participants. Do they agree with the planning process? If not, what don't they like and how could it be done better?

In large, ongoing planning processes (such as strategic planning, business planning, project planning, etc.), it's critical to collect this kind of feedback regularly.

During regular reviews of implementation of the plan, assess if goals are being achieved or not. If not, were goals realistic? Do responsible parties have the resources necessary to achieve the goals and objectives?

Should goals be changed? Should more priority be placed on achieving the goals? What needs to be done?

Write down how the planning process could have been done better. File it away and read it the next time you conduct the planning process.

vii. Recurring Planning Process is at Least as Important as Plan Document

Far too often, primary emphasis is placed on the plan document.

This is extremely unfortunate because the real treasure of planning is the planning process itself.

During planning, planners learn a great deal from ongoing analysis, reflection, discussion, debates and dialogue around issues and goals in the system.

The ongoing communications are what sensitize people to understanding and following the values and behaviors described

viii. Nature of the Process Should Be Compatible to Nature of Planners

A prominent example of this type of potential problem is when planners don't prefer the "top down" or "bottom up", "linear" type of planning

For example, going from general to specific along the process of an environmental scan, SWOT analysis, mission/vision/values, issues and goals, strategies, objectives, timelines, etc.

LECTURE # 27

5. ORGANIZATION

5.1 Basic Definition

Basically, an organization is a group of people intentionally organized to accomplish an overall, common goal or set of goals. Business organizations can range in size from two people to tens of thousands.

How you interpret each of the above major parts of an organization depends very much on your values and your nature. People can view organizations as machines, organisms, families, groups, etc.

People are managed through an organizational structure. This hierarchical structure is based on the four cornerstones of management:

- Delegation
- Authority
- Responsibility
- Supervision.

Delegation bestows authority, and authority produces (and requires) responsibility. Both authority and responsibility require supervision, and effective supervision requires a suitable organizational structure:

Most projects are organized as teams, with each team assigned specific functions within the project.

Different types of project require different types of team structure, as for example a team of junior programmers requires a technical team leader while a team of experts may require only an administrative team leader.

It is the project manager's responsibility to select the structure best suited for the project.

Basically an organization is a group of people intentionally organized to accomplish an overall, common goal or set of goals. Business organizations can range in size from two people to tens of thousands.

How you interpret each of the above major parts of an organization depends very much on your values and your nature. People can view organizations as machines, organisms, families, groups, etc.

5.2 Organization as a System

It helps to think of organizations are systems.

Simply put, a system is an organized collection of parts that are highly integrated in order to accomplish an overall goal.

The system has various inputs which are processed to produce certain outputs, which together, accomplish the overall goal desired by the organization.

There is ongoing feedback among these various parts to ensure they remain aligned to accomplish the overall goal of the organization. There are several classes of systems, ranging from very simple frameworks all the way to social systems, which are the most complex. Organizations are, of course, social systems.

Systems have inputs, processes, outputs and outcomes. To explain, inputs to the system include resources such as raw materials, money, technologies and people.

These inputs go through a process where they're aligned, moved along and carefully coordinated, ultimately to achieve the goals set for the system. Outputs are tangible results produced by processes in the system, such as products or services for consumers.

Another kind of result is outcomes, or benefits for consumers, e.g., jobs for workers, enhanced quality of life for customers, etc. Systems can be the entire organization, or its departments, groups, processes, etc.

Feedback comes from, e.g., employees who carry out processes in the organization, customers/clients using the products and services, etc.

Feedback also comes from the larger environment of the organization, e.g., influences from government, society, economics, and technologies.

Each organization has numerous subsystems, as well. Each subsystem has its own boundaries of sorts, and includes various inputs, processes, outputs and outcomes geared to accomplish an overall goal for the subsystem.

Common examples of subsystems are departments, programs, projects, teams and processes to produce products or services, etc.

Organizations are made up of people - who are also systems of systems of systems - and on it goes. Subsystems are organized in a hierarchy needed to accomplish the overall goal of the overall system.

The organizational system is defined by, e.g., its legal documents (articles of incorporation, by laws, roles of officers, etc.), mission, goals and strategies, policies and procedures, operating manuals, etc.

The organization is depicted by its organizational charts, job descriptions, marketing materials, etc.

The organizational system is also maintained or controlled by policies and procedures, budgets, information management systems, quality management systems, performance review systems, etc.

5.3 Structural Dimensions

The organization's structure, or design, is the overall arrangement of the organization's various roles, processes and their relationships in the organization. The design of an organization is a means to accomplishing the organization's overall goal - the structure is not an end in itself.

In systems theory terms, the design ensures that the appropriate inputs go through the necessary processes to produce the required outputs to produce the intended outcomes.

- ***Centralization*** -the extent to which functions are dispersed in the organization, either in terms of integration with other functions or geographically
- ***Formalization*** - regarding the extent of policies and procedures in the organization
- ***Hierarchy*** - regarding the extent and configuration of levels in the structure
- ***Routinization*** - regarding the extent that organizational processes are standardized
- ***Specialization*** - regarding the extent to which activities are refined
- ***Training*** - regarding the extent of activities to equip organization members with knowledge and skills to carry out their roles.

5.4 Organizational Systems

Project-based organizations are those whose operations consist primarily of projects. These organizations fall into two categories:

– Organizations that derive their revenue primarily from performing projects for others—architectural firms, engineering firms, consultants, construction contractors, government contractors, nongovernmental organizations, etc

– Organizations that have adopted *management by projects*, these organizations

tend to have management systems in place to facilitate project management. For example, their financial systems are often specifically designed for accounting, tracking, and reporting on multiple simultaneous projects.

Non project-based organizations often lack management systems designed to support project needs efficiently and effectively.

The absence of project-oriented systems usually makes project management more difficult.

In some cases, non project-based organizations will have departments or other subunits that operate as project-based organizations with systems to match.

The project management team should be acutely aware of how the organization's systems affect the project.

For example, if the organization rewards its functional managers for charging staff time to projects, then the project management team may need to implement controls to ensure that assigned staff members are being used effectively on the project.

5.5 Organizational Cultures and Styles

Most organizations have developed unique and describable cultures. These cultures are reflected in their:

- *Shared values,*
- *Norms,*
- *Beliefs*
- *Expectations*
- *Policies and*
- *Procedures*
- *View of authority relationships and numerous other factors*

Organizational cultures often have a direct influence on the project.

- A team proposing an unusual or high-risk approach is more likely to secure approval in an aggressive or entrepreneurial organization.

- A project manager with a highly participative style is apt to encounter problems in a rigidly hierarchical organization, while a project manager with an authoritarian style will be equally challenged in a participative organization.

The structure of the performing organization often constrains the availability of or terms under which resources become available to the project.

Organizational structures can be characterized as spanning a spectrum from functional to projectized, with a variety of matrix structures in between.

The following table shows key project related characteristics of the major types of enterprise organizational structures:

Table: Organizational Structure Influences on Projects

Project Characteristics	Organization Structure	Functional	Matrix			Projectized
			Weak	*Balanced*	*Strong*	
Project Manager's Authority		Little or non	Limited	Low to Moderate	Moderate to High	High to almost Total
Percent of Performing Organization's Personnel assigned Full time to Project work		Virtually None	0-25%	15-60%	50-95%	85-100%
Project Manager's Role		Part-time	Part time	Full-time	Full-time	Full-time
Common Titles for Project Manager's Role		Project Coordinator/ Project Leader	Project Coordinator/ Project Leader	Project Manager/Project Officer	Project Manager/ Program Manager	Project Manager/ Program Manager
Project Management Administrative Staff		Part-time	Part time	Part-time	Full-time	Full-time

5.6 Traditional Structures of Business Organization

i. *Functional Structure*

Most business organizations start out with a functional structure, or a small variation of this structure. This is the basic "building block" for other structures. In this structure, there is a central office which oversees various departments or major functions, e.g., human resources, finances, sales, marketing, engineering, etc.

Think of a picture that has a box at the top labeled "Central Office". Think of a row of boxes underneath the top box. Each box is labeled, e.g., sales, engineering, human resources, etc.

Connect the boxes with lines coming down from the top box to each of the boxes below. Use **functional structures** when the organization is small, geographically centralized, and provides few goods and services.
When the organization experiences bottlenecks in decision making and difficulties in coordination, it has outgrown its functional structure.

The classic functional organization is a hierarchy where each employee has one clear superior. Staff members are grouped by specialty, such as production, marketing, engineering, and accounting at the top level, with engineering further subdivided into functional organizations that support the business of the larger organization (e.g., mechanical and electrical).

Functional organizations still have projects, but the perceived scope of the project is limited to the boundaries of the function: the engineering department in a functional organization will do its work independent of the manufacturing or marketing departments.

For example, when a new product development is undertaken in a purely functional organization, the design phase is often called a *design project* and includes only engineering department staff. If questions about manufacturing arise, they are passed up the hierarchy to the department head, who consults with the head of the manufacturing department. The engineering department head then passes the answer back down the hierarchy to the engineering project manager.

ii. ***Projectized Structure***

In this structure, there is a centralized corporate office and under it, are various divisions each of which is dedicated to producing and / or selling a certain type of business or product, e.g., product 1, product 2, etc.

Each division that is dedicated to a certain business or product is, in turn, is organized as its own functional structure.

So, for example, the division dedicated to making product 1 has its own sales department, human resources, etc. Basically, project structure is a bunch of functional structures each of which reports to one central office.

Use a divisional structure when the organization is relatively large, geographically dispersed, and/or produces wide range of goods/services.

In a projectized organization, team members are often collocated. Most of the organization's resources are involved in project work, and project managers have a great deal of independence and authority.

Projectized organizations often have organizational units called departments, but these groups either report directly to the project manager or provide support services to the various projects.

iii. ***Matrix Structure***

Think of the functional structure. Imagine if you took someone from each of the major functions in the functional structure (the boxes along the bottom of the organization chart), e.g., people from sales, engineering, etc., and organized them into a separate group intended to produce and sell one certain kind of product or service.

Members of this group stay together until that product is produced or they continue to sell and service it. This overall structure (made up of a functional structure that also has groups assigned to products) is a matrix structure.

This structure is useful because it focuses highly skilled people from across the organization to work on a complex product or service.

It can be difficult, though, because each person essentially reports to two supervisors: the supervisor of the functional area (e.g., engineering) and the product manager, as well.

When the organization needs constant coordination of its functional activities, then lateral relations do not provide sufficient integration. Consider the matrix structure.

To adopt the matrix structure effectively, the organization should modify many traditional management practices.

Matrix organizations are a blend of functional and projectized characteristics.

Weak matrices maintain many of the characteristics of a functional organization, and the project manager role is more that of a coordinator or expediter than that of a manager.

In similar fashion, strong matrices have many of the characteristics of the projectized organization—full-time project managers with considerable authority and full-time project administrative staff.

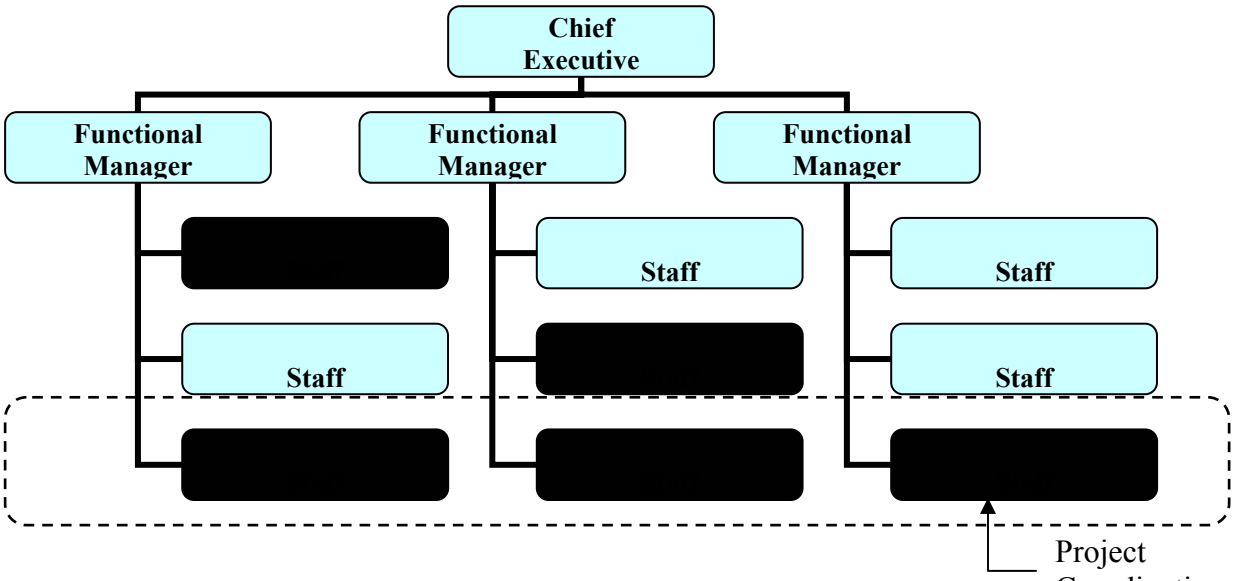


Figure 1: Weak Matrix Organization

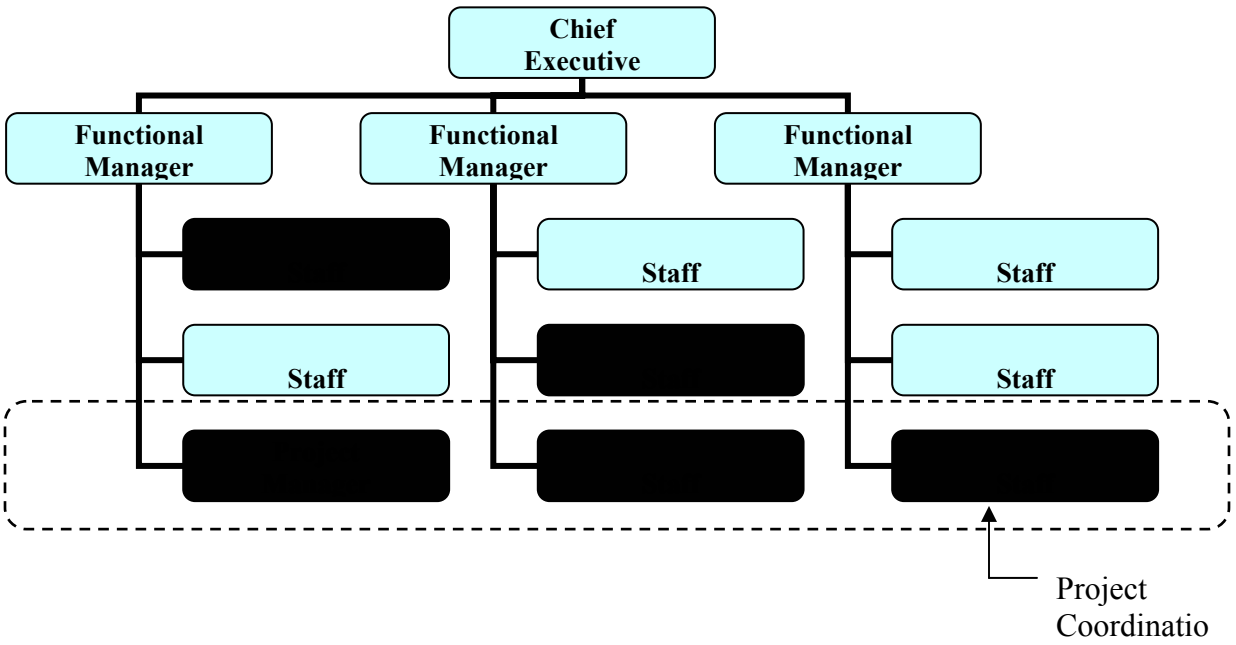


Figure 2: Balanced Matrix Organization

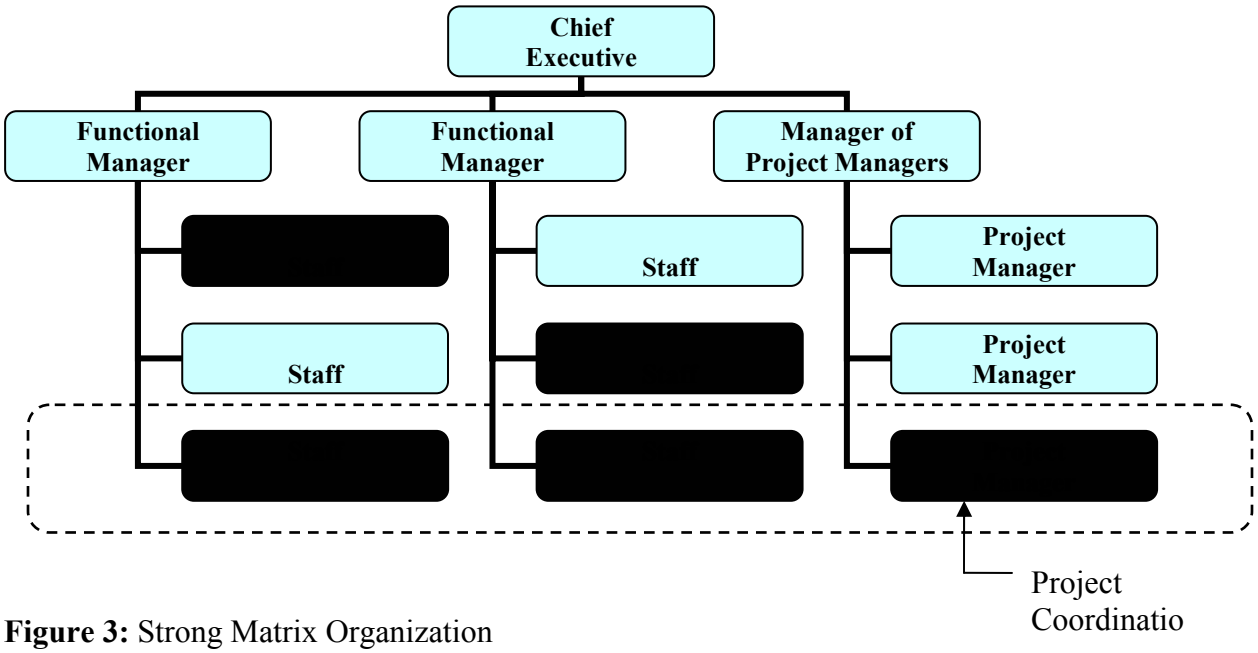


Figure 3: Strong Matrix Organization
(Black boxes represent staff engaged in project activites)

iv. Project Office

There is a range of uses for what constitutes a project office. A project office may operate on a continuum from providing support functions to project managers in the form of training, software, templates, etc. to actually being responsible for the results of the project.

Most modern organizations involve all these structures at various levels. For example, even a fundamentally functional organization may create a special project team to handle a critical project.

Such a team may have many of the characteristics of a project in a projectized organization.

The team may include full-time staff from different functional departments, it may develop its own set of operating procedures, and it may operate outside the standard, formalized reporting structure.

LECTURE # 28

5. ORGANIZATION

5.7 ORGANIZATIONAL PLANNING

Organizational planning involves identifying, documenting, and assigning project roles, responsibilities, and reporting relationships.

Roles, responsibilities, and reporting relationships may be assigned to individuals or to groups. The individuals and groups may be part of the organization performing the project, or they may be external to it. Internal groups are often associated with a specific functional department such as engineering, marketing, or accounting.

On most projects, the majority of organizational planning is done as part of the earliest project phases.

However, the results of this process should be reviewed regularly throughout the project to ensure continued applicability. If the initial organization is no longer effective, then it should be revised promptly.

Organizational planning is often tightly linked with communications planning since the project's organizational structure will have a major effect on the project's communications requirements.

5.7.1 Inputs to Organizational Planning

i. **Project interfaces.** Project interfaces generally fall into one of three categories:

- ***Organizational interfaces***—formal and informal reporting relationships among different organizational units. Organizational interfaces may be highly complex or very simple. For example, developing a complex telecommunications system may require coordinating numerous subcontractors over several years, while fixing a programming error in a system installed at a single site may require little more than notifying the user and the operations staff upon completion.

- ***Technical interfaces***—formal and informal reporting relationships among different technical disciplines. Technical interfaces occur both within project phases (e.g., the site design developed by the civil engineers must be compatible with the superstructure developed by the structural

engineers) and between project phases (e.g., when an automotive design team passes the results of its work along to the retooling team that must create the manufacturing capability for the vehicle).

- ***Interpersonal interfaces***—formal and informal reporting relationships among different individuals working on the project. These interfaces often occur simultaneously, as when an architect employed by a design firm explains key design considerations to an unrelated construction contractor's project management team.

ii. **Staffing requirements.** Staffing requirements define what kinds of competencies are required from what kinds of individuals or groups and in what time frames. Staffing requirements are a subset of the overall resource requirements identified during resource planning.

iii. **Constraints**. Constraints are factors that limit the project team's options. A project's organizational options may be constrained in many ways. Common factors that may constrain how the team is organized include, but are not limited to, the following:

- ***Organizational structure of the performing organization***—an organization whose basic structure is a strong matrix means a relatively stronger role for the project manager than one whose basic structure is a weak matrix.
- ***Collective bargaining agreements***—contractual agreements with unions or other employee groups may require certain roles or reporting relationships (in essence, the employee group is a stakeholder).
- ***Preferences of the project management team***—if members of the project management team have had success with certain structures in the past, then they are likely to advocate similar structures in the future.
- ***Expected staff assignment*** —how the project is organized is often influenced by the competencies of specific individuals.

5.7.2 Tools and Techniques for Organizational Planning

i. **Templates.** Although each project is unique, most projects will resemble another project to some extent. Using the role and responsibility definitions or reporting relationships of a similar project can help expedite the process of organizational planning.

ii. **Human resource practices**. Many organizations have a variety of policies, guidelines, and procedures that can help the project management team with various aspects of organizational planning. For example, an organization that views managers as "coaches" is likely to have documentation on how the role of "coach" is to be performed.

iii. **Organizational theory**. There is a substantial body of literature describing how organizations can and should be structured. Although only a small subset of this body of literature is specifically targeted toward project organizations, the project management team should be generally familiar with the subject of organizational theory so as to be better able to respond to project requirements.

iv. **Stakeholder analysis**. The identification of stakeholders and the needs of the various stakeholders should be analyzed to ensure that their needs will be met.

5.7.3 Outputs from Organizational Planning

i. **Role and responsibility assignments**. Project roles (who does what) and responsibilities (who decides what) must be assigned to the appropriate project stakeholders. Roles and responsibilities may vary over time. Most roles and responsibilities will be assigned to stakeholders who are actively involved in the work of the project, such as the project manager, other members of the project management team, and the individual contributors. The roles and responsibilities of the project manager are generally critical on most projects, but vary significantly by application area. Project roles and responsibilities should be closely linked to the project scope definition. A Responsibility Assignment Matrix is often used for this purpose. On larger projects, RAM s may be developed at various levels. For example, a high-level RAM may define which group or unit is responsible for each component of the work breakdown structure, while lower-level RAM s are used within the group to assign roles and responsibilities for specific activities to particular individuals.

ii. **Staffing management plan.** The staffing management plan describes when and how human resources will be brought onto and taken off of the project team. The staffing plan may be formal or informal, highly detailed or broadly framed, based on the needs of the project. It is a subsidiary element of the overall project plan. The staffing management plan often includes resource histograms. Particular attention should be paid to how project team members (individuals or groups) will be released when they are no longer needed on the project. Appropriate reassignment procedures may:

   - **Reduce costs** by reducing or eliminating the tendency to "make work" to fill the time between this assignment and the next.
   - **Improve morale** by reducing or eliminating uncertainty about future employment opportunities.

iii. **Organization chart.** An organization chart is any graphic display of project reporting relationships. It may be formal or informal, highly detailed or broadly framed, based on the needs of the project. For example, the organization chart for a three- to four-person internal service project is unlikely to have the rigor and detail of the organization chart for a 3,000-person disaster response team. An

Organizational Breakdown Structure (OBS) is a specific type of organization chart that shows which organizational units are responsible for which work packages.

iv. **Supporting detail**. Supporting detail for organizational planning varies by application area and project size. Information frequently supplied as supporting detail includes, but is not limited to: Organizational impact—what alternatives are precluded by organizing in this manner.

- ***Job descriptions***—written outlines by job title of the competencies, responsibilities, authority, physical environment, and other characteristics involved in performing a given job. Also called position descriptions.
- ***Training needs***—if the staff to be assigned is not expected to have the competencies needed by the project, those competencies will need to be developed as part of the project.

5.8 STAFF ACQUISITION

Staff acquisition involves getting the needed human resources (individuals or groups) assigned to and working on the project. In most environments, the "best" resources may not be available, and the project management team must take care to ensure that the resources that are available will meet project requirements.

5.8.1 Inputs to Staff Acquisition

i. ***Staffing management plan.*** It includes the project's staffing requirement

ii. ***Staffing pool description***. When the project management team is able to influence or direct staff assignments, it must consider the characteristics of the potentially available staff. Considerations include, but are not limited to:

- Previous experience—have the individuals or groups done similar or related work before? Have they done it well?
- Personal interests—are the individuals or groups interested in working on this project?
- Personal characteristics—are the individuals or groups likely to work well together as a team?
- Availability—will the most desirable individuals or groups be available in the necessary time frames?
- Competencies and proficiency—what competencies are required and at what level?

iii. ***Recruitment practices***. One or more of the organizations involved in the project may have policies, guidelines, or procedures governing staff assignments. When they exist, such practices act as a constraint on the staff-acquisition process.

5.8.2 Tools and Techniques for Staff Acquisition

i. **Negotiations.** Staff assignments must be negotiated on most projects. For example, the project management team may need to negotiate with:

- Responsible functional managers to ensure that the project receives appropriately competent staff in the necessary time frame.
- Other project management teams within the performing organization to assign scarce or specialized resources appropriately.

The team's influencing competencies play an important role in negotiating staff assignments, as do the politics of the organizations involved. For example, a functional manager may be rewarded based on staff utilization. This creates an incentive for the manager to assign available staff who may not meet all of the project's requirements.

ii. **Pre-assignment.** In some cases, staff may be pre-assigned to the project. This is often the case when a) the project is the result of a competitive proposal, and specific staff was promised as part of the proposal, or b) the project is an internal service project, and staff assignments were defined within the project charter.

iii. **Procurement.** Project procurement management can be used to obtain the services of specific individuals or groups of individuals to perform project activities. Procurement is required when the performing organization lacks the in-house staff needed to complete the project (e.g., as a result of a conscious decision not to hire such individuals as full-time employees, as a result of having all appropriately competent staff previously committed to other projects, or as a result of other circumstances).

5.8.3 Outputs from Staff Acquisition

i. **Project staff assigned**. The project is staffed when appropriate people have been reliably assigned to work on it. Staff may be assigned full time, part time, or variably, based on the needs of the project.

ii. **Project team directory**. A project team directory lists all the project team members and other stakeholders. The directory may be formal or informal, highly detailed or broadly framed, based on the needs of the project.

5.9 TEAM DEVELOPMENT

Team development includes both enhancing the ability of stakeholders to contribute as individuals as well as enhancing the ability of the team to function as a team. Individual development (managerial and technical) is the foundation necessary to develop the team. Development as a team is critical to the project's ability to meet its objectives.

Team development on a project is often complicated when individual team members are accountable to both a functional manager and the project manager. Effective management of this dual reporting relationship is often a critical success factor for the project, and is generally the responsibility of the project manager. Team development occurs throughout the project.

5.9.1 Inputs to Team Development

i. **Project staff**. The staff assignments implicitly define the individual competencies and team competencies available upon which to build.

ii. **Project plan.** The project plan describes the technical context within which the team operates.

iii. **Staffing management plan.**

iv. **Performance reports.** Performance reports provide feedback to the project team about performance against the project plan.

v. **External feedback**. The project team must periodically measure itself against the expectations of those outside the project.

5.9.2 Tools and Techniques for Team Development

i. **Team-building activities**. Team-building activities include management and individual actions taken specifically and primarily to improve team performance. Many actions—such as involving non-management-level team members in the planning process, or establishing ground rules for surfacing and dealing with conflict— may enhance team performance as a secondary effect. Team-building activities can vary from a five-minute agenda item in a regular status review meeting to an extended, off-site, professionally facilitated experience designed to improve interpersonal relationships among key stakeholders. There is a substantial body of literature on team building. The project management team should be generally familiar with a variety of team-building activities.

ii. **General management skills.** General management skills are of particular importance to team development.

iii. **Reward and recognition systems.** Reward and recognition systems are formal management actions that promote or reinforce desired behavior. To be effective, such systems must make the link between project performance and reward clear, explicit, and achievable. For example, a project manager who is to be rewarded for meeting the project's cost objective should have an appropriate level of control over staffing and procurement decisions. Projects must often have their own reward and recognition systems since the systems of the performing

organization may not be appropriate. For example, the willingness to work overtime to meet an aggressive schedule objective *should* be rewarded or recognized; needing to work overtime as the result of poor planning *should not* be. Reward and recognition systems must also consider cultural differences. For example, developing an appropriate team reward mechanism in a culture that prizes individualism may be very difficult.

iv. **Co-location**. Collocation involves placing all, or almost all, of the most active project team members in the same physical location to enhance their ability to perform as a team. Collocation is widely used on larger projects and can also be effective for smaller projects (e.g., with a *war room*, where the team congregates and posts schedules, updates, etc.). On some projects, collocation may not be an option; where it is not viable, an alternative may be scheduling frequent face to face meetings to encourage interaction.

v. **Training.** Training includes all activities designed to enhance the competencies of the project team. Some authors distinguish among training, education, and development, but the distinctions are neither consistent nor widely accepted. Training may be formal (e.g., classroom training, computer-based training) or informal (e.g., feedback from other team members). There is a substantial body of literature on how to provide training to adults. If the project team members lack necessary management or technical skills, such skills must be developed as part of the project, or steps must be taken to re-staff the project appropriately. Direct and indirect costs for training are generally paid by the performing organization.

5.9.3 Outputs from Team Development

i. **Performance improvements.** Team performance improvements can come from many sources and can affect many areas of project performance; for example:

- Improvements in individual skills may allow a specific person to perform assigned activities more effectively.
- Improvements in team behaviors (e.g., surfacing and dealing with conflict) may allow project team members to devote a greater percentage of their efforts to technical activities.
- Improvements in either individual or team competencies may facilitate identifying and developing better ways of doing project work.

ii. **Input to performance appraisals.** Project staff should generally provide input to the appraisals of any project staff members with whom they interact in a significant way.

5.10 Organizational Management Tools

i. **Management Development**
   - Responsibility

- Authority
- Competence
- Resource Distribution
- Pre-requisites
- Constraints
- Calendar

ii. Supervisory Training

- Field/on site operations
- Concept clearance
- Procedural details
- Resource management
- Activity Scheduling

iii. Team Building

- Managers
- Professionals
- Technical support group
- Logistical support group
- Skill set evaluation

iv. Vital Statistics

- Historical facts
- Technical data
- Direct concerns
- Lesson learned
- Identification of missing links
- Reliability of data
- Relevance of data

v. Progress reporting

- Mandatory periodic reports
- Exception reports
- Event reporting
- Current status reporting
- Reporting formats
- Reporting frequency
- Report recipient
- Reporting officer
- Reporting Media
- Response analysis (of previous reporting)
- Review of Reports
- Signing of Reports
- Tracking of Reports
- Mitigations offered
- Corresponding the dead lines

vi. Compliance of Rule of Business

- Financial
- Procedural
- Administrative
- Justifications for deviations
- Acceptance of derailments

vii. Trade Offs

- Trade off between DO or DON'T
- Quality of job (in the light of constraints)
- Limiting the scope/deliverables
- Meeting targets with minimum standards
- Unavoidable mandatory deliverables

viii. Quality Assurance

- Standard QA procedures
- Self defined measures
- Task-specific controls

ix. Beneficiaries Concern

- Acceptance
- Enthusiasm
- Adding comforts in terms of use, cost or time
- Confidence building-The Reliability

5.11 Contractual Obligations

A contract is a mutually binding agreement that obligates the seller to provide the specified product and obligates the buyer to pay for it.

*A contract is a legal relationship subject to remedy in the court*s. The agreement may be simple or complex, usually (but not always) reflecting the simplicity or complexity of the product.

Contracts may be called, among other names, a *contrac*t, an *agreemen*t, a *subcontrac*t, a *purchase orde*r, or a *memorandum of understandin*g. Most organizations have documented policies and procedures specifically defining who can sign such agreements on behalf of the organization, typically called a *delegation of procurement authorit*y.

Although all project documents are subject to some form of review and approval, the legally binding nature of a contract usually means that it will be subjected to a more extensive approval process. In all cases, a primary focus of the review and approval process should be to ensure that the contract language describes a product or service that will satisfy the identified need. In the case of major

projects undertaken by public agencies, the review process may even include public review of the agreement.

- **Types of Contracts**

Owing to the rapid advances in technology during the last several decades, it has become increasingly necessary for high technology organizations to specialize in specific, well-defined areas. Specialization has not only defined many new branches of engineering, it has also defined areas of expertise within the engineering disciplines. This is especially true of software engineering.

Frequently, organizations that do not specialize in software development hire other organizations that do so to develop software for them. Even organizations that do develop their own software may decide to hire outside specialists in specific areas. IBM hired Microsoft to develop the PC-DOS operating system, because Microsoft had experience in developing microcomputer systems and IBM had not.

The development of software is much less deterministic and more risky than other areas of technology. This often leads to misunderstandings and disagreements that could have been avoided if they had been anticipated and contained early enough.

To standardize our terminology, the organization to whom, a proposal is being submitted will be referred to as the customer, and the organization submitting the proposal will be referred to as the pro-poser. Other terms commonly used elsewhere for the pro-poser include bidder, vendor or contractor and for the customer requestor or issuer. The organization submitting the winning proposal, after being selected, will be referred to as the developer.

a) The cost-plus contract

Cost-plus is a contractual relationship where the developer is paid for the cost of the service provided and in addition is allowed an agreed profit margin.

This is rather like renting a car the customer pays for the time that the car is used (by the hour, day, week etc.), and for any other expenses such as insurance and gasoline. Thus, in a cost-plus contract the total cost of the projects is only known after the project has been completed.

As an example, company Alpha may contract software company Beta to develop a system, company Beta will be paid $80 by company Alpha for each hour invested by their engineers in the project. In additional 20 percent may then be added to cover managerial, secretarial and other services. Additional expenses incurred by company Beta for the benefit of the project would then be reimbursed by company Alpha. These expenses might cover such areas as:

- Special purpose development equipment (computers, compilers, networks etc.)
- Travel expenses incurred by employees of company Beta for the benefit of the project
- Target equipment procured by company Beta for the use of company Alpha
- Services from other outside sources requested by company Beta for the project

The customer company, Alpha, may require the developer, company Beta, to receive prior authorization before incurring any single expense exceeding $250, and any expense in excess of $6000 monthly total. Such authorization should always be in writing. This defines a basic cost-plus contractual relationship between the two companies.

In many cases, cost-plus can be the most appropriate way to contract development work however, there are numerous potential problems. A conflict of interest may arise due to the developer's lack of motivation to complete the project as quickly as possible, or due to the customer's reluctance to authorize additional expenses.

Cost-plus is often appropriate for small undefined projects, when it is difficult to identify the project's requirements in advance. In fact, in many cases the requirements phase of a project is offered as a cost-plus contract, all the remaining phases are contracted at a fixed price.

The requirements phase is then used to bring the rest of the project to a sufficiently well-defined state from which it can then be contracted at a fixed price. Occasionally, one company is awarded a cost-plus contract for the requirements phase, and another company is awarded the remaining phases as a fixed price contract.

Cost-plus may be preferred by the customer who wants to retain control of the development process. In some cases, the developer is perceived as an extension of the customer's organization, and the development activities are managed by the customer. A cost-plus contract should cover the following issues:

- List of persons to be assigned to the project
- Work definition
- The assignment percentage for each person
- Hourly or daily work rate for each person
- Administrative overhead
- Authorized expenses to be reimbursed
- Billing procedure
- Payment procedure
- Termination procedure

The assignment percentage refers to the amount of time each person will be expected to devote to the project. This may be 100 percent for some engineers, and 50-60 percent for experts in specific areas.

The assignment percentage may also be quoted in terms of maximum or minimum, meaning that, for example, a quality assurance engineer will devote no more than 20 hours a week to the project, and no fewer than 10 hours a week to the project.

The billing rate may be a fixed rate for all persons assigned to the project, or individual rates may be set for each person or class of people.

For example, for each hour worked on the project, the developer will bill $80, irrespective of who, worked that hour. Or the contract may stipulate that design engineers bill at $120 per hour, coders at $60 per hour, documentation writers at $50 per hour, and so forth. The most difficult cost-plus contract billing rate method is the individual billing method, where Frank Jones is billed at $90 per hour, John Smith at $75 etc. This means that each time a person is replaced or added to the project, the hourly rate must renegotiated.

For a software development organization, there can be real advantages in cost-plus contracts these include:

- No financial or business risk
- Acquisition of knowledge and experience at the expense of another organization

However, as in most cases, these advantages come with some disadvantages, which include:

- Low business profit
- Possible staff discontent
- Reduced control of staff and development work
- Potential friction with the customer due to a lack of well-defined goals and motivation factors
- Contract continuity is not assured

Most employees prefer a clear definition of the hierarchy to which they belong. In a cost-plus contract the employee works within the customer's hierarchy, but belongs to the developer's hierarchy, and this can cause discontent.

In general, from the developer's perspective, a cost-plus contract is a solid, low profit, no risk business relationship.

From the customer's perspective, the advantages of a cost-plus contract are:

- Retention of control over development
- No commitment needed for a full project contract
- A possible reduced business risk (due to the ability to terminate the contract at any time

The customer's possible disadvantages are:

- Increased development costs
- Customer's assumption of development risks
- Increased involvement in development
- Potential friction with the developer due to a lack of well-defined goals and motivation factors

For the customer, the desirability of a cost-plus contract is difficult to establish.

Clearly is dependent on the type of project and the conditions under which it will be developed, as we as on other non-technical business considerations.

b) The fixed price contract

A fixed price contract is a commitment by the developer to provide an agreed product or service for an agreed fee, within an agreed schedule.

This is similar to purchasing a bus ticket, when the bus company agrees to take the customer to a specific destination within a published timetable, and for an agreed fee. Of course, travelers can elect to rent a car, instead of purchasing a bus ticket, and then drive to their destinations themselves. However, this may turn out to be more expensive, and requires of the traveler some prior skills and knowledge, such driving skills and knowledge of the route to the destination. So travelers (or customers) must decide between providing the service themselves and contracting someone else to provide the service.

A fixed price contract can only be applied to a well-defined project. Both customer and developer must be able to define the final deliverable product or service. Once this has been achieved, one of the main weaknesses of the fixed price contract will have been removed. The advantages of a fixed price contract for the developer include:

- Full control of the development process
- Possible higher business profit
- Commitment for a complete project

The commitment for a complete project is a significant advantage over cost-plus contract that may end at any time, at the customer's discretion. Of course, fixed price contracts also have some disadvantages for the developer, which include:

- Assumption of business and development risks
- Potential friction with the customer due to:

  - continuing requirement changes
  - project completion criteria
  - interpretation of requirements

A successful software organization will often prefer a fixed price contract. These are usually the projects that build a company's professional reputation, and generate profit to enable growth.

Unfortunately, these are also the projects that generate loss, and which often severely harm a company. Stiff competition for an important contract occasionally tempts a company to underbid, which ultimately generates losses for the developer.

It is almost inevitable in any project that the developer will be requested to change requirements during development. Such changes are usually associated with additional cost the customer, and are invariably a cause of disagreement between developer and customer.

This is often due to unclear or ambiguous requirements, which, in turn lead to disagreement regarding the criteria for project completion. This, essentially, returns the contract to insufficiently defined state.

From the customer's perspective, the advantages in a fixed price contract include:

- A fixed budget for the project.
- Most of the development risks are transferred to the developer
- Minimal involvement in the development process

The disadvantages to the customer are:

- Risk of late delivery by the developer
- Reduced control of the development process
- Potential friction with the developer due to:

  - high cost of requirement changes
  - unclear project completion criteria
  - interpretation of requirements

Even though the interests of the developer and the customer may be different, fixed price contracts are still often preferred by both parties. If the project is sufficiently detailed and clear and if the relationship between the two parties is

well defined, then fixed price contracts can be beneficial to both the developer and the customer.

- **The Cost-Plus Vs Fixed Price**

  There is often a real or imagined conflict of interest between the developer and the customer. The customer wants to spend less and the developer wants to earn more. As we shall see, a good relationship between developer and customer need not necessarily lead to this conflict of interest.

  There are basically two types of contractual relationship between the customer and the Developer:

  - **Cost-plus** (also called Time and material)
  - **Fixed price**

  Most other relationships are some kind of combination of these two

- **Contract type selection:**

  Different types of contracts are more or less appropriate for different types of purchases. Contracts generally fall into one of three broad categories:

  - **Fixed-price or lump-sum contracts**—this category of contract involves a fixed total price for a well-defined product. To the extent that the product is not well defined, both the buyer and seller are at risk—the buyer may not receive the desired product or the seller may need to incur additional costs to provide it. Fixed-price contracts may also include incentives for meeting or exceeding selected project objectives, such as schedule targets.

  - **Cost-reimbursable contracts**—this category of contract involves payment (Reimbursement) to the seller for its actual costs plus, typically, a fee representing seller profit. Costs are usually classified as *direct costs* or *indirect cost*s. Direct costs are costs incurred for the exclusive benefit of the project (e.g. salaries of full-time project staff). Indirect costs, also called overhead costs, are costs allocated to the project by the performing organization as a cost of doing business (e.g., salaries of corporate executives). Indirect costs are usually calculated as a percentage of direct costs. Cost-reimbursable contracts often include incentives for meeting or exceeding selected project objectives, such as schedule targets or total cost.

  - **Time and Material (T&M) contracts**—T&M contracts are a hybrid type of contractual arrangement that contains aspects of both cost-reimbursable and fixed-price-type arrangements. T&M contracts resemble cost-type arrangements in that they are open ended, because the full value of the arrangement is not defined at the time of the award. Thus, T&M contracts can

grow in con-tract value as if they were cost-reimbursable-type arrangements. Conversely, T&M arrangements can also resemble fixed-unit arrangements when, for example, the unit rates are preset by the buyer and seller, as when both parties agree on the rates for the category of "senior engineers."

- **Other Customer-Developer Relationships**

  Cost-plus and fixed price are two of the traditional contractual relationships between developer and customer. There are many variations of these two basic relationships, including various combinations that are tailored to suit specific projects. Some of these relationships are associated with the roles of customer and developer, and attempt to provide more incentives for the developer to support the customer's objectives beyond contractual obligations. Additional types of customer-developer relationship include:

  - Combinations of fixed price and cost-plus
  - Joint ventures
  - Royalty agreements
  - Long-term commitments

  Joint ventures are instances where the customer-developer dividing line can become hazy, and many of the previously discussed advantages and disadvantages may not apply. There are many cases where some form of joint venture may be desirable for both parties, such as when the developer wants to retain rights to the product, or when the developer joins the customer in funding part of the development effort.

One way the customer can offer the developer moderate participation in the business aspect of the project is by substituting royalties as partial payment. This generates an added dimension to the developer's interest in the success of the project. The royalties are usually such that the failure of the project would produce less revenue for the developer than a straightforward fixed price contract, and the success of the project will increase the developer's revenue.

Long-term relationships are often important for the developer. In many cases, long term commitments are also in the customer's interest. This occurs when the developer, by being awarded the initial contract, gains, through acquired knowledge, a major advantage over others for subsequent development work. Clearly, when the developer successfully completes a large and complex project, a significant advantage is then acquired over other companies with respect to future extensions of the project. A long-term commitment may then be of mutual interest to both parties, wherein the customer assures future services from the developer and the developer assures a long term income commitment.

5.12 The statement of work (SOW)

The statement of work is the basis of the contract between the pro-poser and the customer, and is often incorporated into the contract. The SOW contains a detailed list of all work to be performed by the pro-poser for the benefit of the customer.

It is a narrative description of *products* or *services* to be supplied by the project. For internal projects, the project *initiator* or *sponsor* provides the statement of work based on business needs, or product or service *requirement*s.

For external projects, the statement of work can be received from the *customer* as part of a bid document, for example, *request for proposa*l, *request for informatio*n, request for bid, or as part of a *contrac*t. The SOW indicates a:

- ***Business need*** - an organization's business need, can be based on needed training, market demand, technological advance, legal requirement, or governmental standard.

- ***Product scope description*** - documents the product requirements and characteristics of the product or service that the project will be undertaken to create. The product requirements will generally have less detail during the initiation process and more detail during later processes, as the product characteristics are progressively elaborated. These requirements should also document the relationship among the products or services being created and the business need or other stimulus that causes the need. While the form and substance of the product requirements document will vary, it should always be detailed enough to support later project planning.

- ***Strategic plan*** - all projects support the organization's strategic goals—the strategic plan of the performing organization should be considered as a factor in project selection decisions.

The SOW starts as a general list of required deliverables in the RFP. A more detailed version t of the SOW is submitted as part of the proposal, and is still considered only an initial description of the work to be performed. The blinding version of the SOW is finalized during contract negotiations, or after the detailed project requirements have been completed.

Following table presents an example of an SOW outline for a software project. The list of items varies considerably, depending on the type of project being developed; for example not all projects include the delivery of hardware components, and not all projects require training or installation.

Table: A sample SOW outline for a software project

14. Referenced documents
- requirements specification
- existing system description
- customer's RFP
- developer's proposal
- vendor's and developer's technical literature

15. Software deliverables
- functionality (as documented in the requirements specification)
- list of major software components

16. Equipment and hardware deliverables
- functionality (as documented in the requirements specification)
- list of major hardware components

17. Training
- user courses
- operator training
- installation training

18. Market research
19. Procurement
20. Supervision of subcontractors
21. Documentation
- development documentation
- user documentation
- maintenance documentation
- other technical documentation

22. Testing
- alpha testing
- beta testing
- acceptance tests (ATP)

23. Installation
24. Maintenance services
25. Other services and deliverable items
26. Method of delivery
- software
- documentation
- hardware

The basic guideline for the preparation of the SOW is that any activity, service or product required by the customer, and agreed to by the developer, must be included. This means that there can be no binding work items that were informally understood or agreed to verbally, which do not appear in the SOW.

The formal SOW must include all and only the work to be performed. This condition prevents misunderstandings and disagreements later, after the project begins.

The statement of work (SOW) describes the procurement item in sufficient detail to allow prospective sellers to determine if they are capable of providing the item. "Sufficient detail" may vary, based on the nature of the item, the needs of the buyer, or the expected contract form.

Some application areas recognize different types of SOW. For example, in some government jurisdictions, the term *SOW* is reserved for a procurement item that is a clearly specified product or service, and the term *Statement of Objectives* (SOO) is used for a procurement item that is presented as a problem to be solved.

The statement of work may be revised and refined as it moves through the procurement process. For example, a prospective seller may suggest a more efficient approach or a less costly product than that originally specified. Each individual procurement item requires a separate statement of work. However, multiple products or services may be grouped as one procurement item with a single SOW.

The statement of work should be as clear, as complete, and as concise as possible. It should include a description of any collateral services required, such as performance reporting or post-project operational support for the procured item.

In some application areas, there are specific content and format requirements for a SOW.

5.12.1 SOW Template

i. ***Scope of work:*** Describe the work to be done in detail. Specify the hardware and software involved and the exact nature of the work.

ii. ***Location of Work:*** Describe where the work must be performed. Specify the location of hardware and software and where the people must perform the work.

iii. ***Period of Performance:*** Specify when the work is expected to start and end, working hours, number of hours that can be billed per week, where the work must be performed, and related schedule information. Optional "Compensation" section.

iv. ***Deliverables Schedule:*** List specific deliverables, describe them in detail, and specify when they are due.

v. ***Applicable Standards:*** Specify any company or industry-specific standards that are relevant to performing the work. Often an Assumptions section as well.

vi. ***Acceptance Criteria:*** Describe how the buyer organization will determine if the work is acceptable

vii. ***Special Requirements:*** Specify any special requirements such as hardware or software certifications, minimum degree or experience level of personnel, travel requirements, documentation, testing, support, and so on.

5.13 Software built around a Scheduling Engine

These packages were originally created to manage one project at a time, but over the years have been enhanced to handle multiple projects.

Well known examples include:

1. **Primavera**
2. **TurboProject**
3. **OpenPlan**
4. **Microsoft Project**
5. **AutoPlan**
6. **Project Scheduler 8**
7. **CA SuperProject**
8. **Timeline**

LECTURE # 29

6. ESTIMATION

6.1 Estimation - Concepts

Watts Humphrey in his book, Managing the Software process, has said, "*If you don't know where you are, a map won't help.*" This saying is very relevant while dealing with software project estimation. In a software project, unless you are sure that your estimation is accurate, you cannot make much progress.

Estimation of factors such as cost, effort, risks, and resources is crucial. It gives you a fair idea of the size of the project. You can use the information about size to estimate the cost, effort, and duration of the project. This further helps plan for resources and schedule the project.

In the early days of computing, software costs constituted a small percentage of the overall computer-based system cost. An order of magnitude error in estimates of software cost had relatively little impact.

Today, software is the most expensive element of virtually all computer-based systems. For complex, custom systems, a large increased cost estimation error can make the difference between profit and loss. Cost overrun can be disastrous for the developer.

Software cost and effort estimation will never be an exact science. Too many variables - human, technical, environmental, political - can affect the ultimate cost of software and effort applied to develop it.

However, software project estimation can be transformed from a black art to a series of systematic steps that provide estimates with acceptable risk.

To achieve reliable cost and effort estimates, a number of options arise:

1. **Delay estimation** until late in the project (obviously, we can achieve *100%* accurate estimates after the project is complete!).
2. **Base estimates** on similar projects that have already been completed.
3. **Use relatively simple decomposition techniques** to generate project cost and effort estimates.
4. **Use one or more empirical models** for software cost and effort estimation.

Unfortunately, the first option, however attractive, is not practical. Cost estimates must be provided 'up front.' However, we should recognize that the longer we

wait, the more we know, and the more we know, the less likely we are to make serious errors in our estimates.

The second option can work reasonably well, if the current project is quite similar to past efforts and other project influences (e.g., the customer, business conditions, the SEE, deadlines) are equivalent. Unfortunately, past experience has not always been a good indicator of future results.

The remaining options are viable approaches to software project estimation. Ideally, the techniques noted for each option should be applied in tandem; each used as a cross-check for the other.

Software project estimation is a form of problem solving, and in most cases, the problem to be solved (i.e., developing a cost and effort estimate for a software project) is too complex to be considered in one piece. For this reason, we decompose the problem, re-characterizing it as a set of smaller (and hopefully, more manageable) problems.

Following are some points related to project estimation:

- Estimation is very difficult to do, but is often needed
- It is created, used or refined during
  - *Strategic planning*
  - *Feasibility study and/or SOW*
  - *Proposals*
  - *Vendor and sub-contractor evaluation*
  - *Project planning (iteratively)*
- Basic process involves:
  - *Estimate the size of the product*
  - *Estimate the effort (man-months)*
  - *Estimate the schedule*
  - **NOTE:** Not all of these steps are always explicitly performed

6.2 Estimation – A Critical factor

In a software project, unless you are sure that your estimation is accurate, you cannot make much progress.

Estimation of following factors are essential:

- *Cost,*
- *Effort,*

- *Risks*
- *Resources*

Estimation gives you a fair idea of the size of the project. You can use the information about size to estimate the cost, effort, and duration of the project. This further helps plan for resources and schedule the project.

Estimation of resources, cost, and schedule for a software engineering effort requires:

- *Experience,*
- *Access to good historical information*
- *Courage to commit to quantitative predictions when qualitative information is all that exists*

Estimation carries inherent risk and this risk leads to uncertainty.

a) Project complexity
b) Project size
c) The degree of structural uncertainty
d) The availability of historical information
e) Risk

a) Project complexity has a strong effect on the uncertainty, inherent in planning. Complexity, however, is a relative measure that is affected by familiarity with past effort. The first-time developer of a sophisticated e-commerce application might consider it to be exceedingly complex. However, a software team developing its tenth e-commerce Web site would consider such work run of the mill. A number of quantitative software complexity measures can be applied as per the need of project. Such measures are applied at the design or code level and are therefore difficult to use during Software planning (before a design and code exist). However, other, more subjective assessments of complexity (e.g., the function point complexity adjustment factors) can be established early in the planning process.

b) Project size is another important factor that can affect the accuracy and efficacy of estimates. As size increases, the interdependency among various elements of the software grows rapidly. Problem decomposition, an important approach to estimating, becomes more difficult because decomposed elements may still be alarming. To paraphrase Murphy's Law: "What can go wrong will go wrong"- and if there are more things that can fail, more things will fail.

c) The degree of structural uncertainty has an effect on estimation risk. In this context, structure refers to the degree to which requirements have been solidified, the ease with which functions can be compartmentalized and the hierarchical nature of the information that must be processed.

d) **The availability of historical information** has a strong influence on estimation risk. By looking back, we can emulate things that worked and improve areas where problems arose.

e) **When comprehensive software** metrics are available for past projects, estimates can be made with greater assurance, schedules can be established to avoid past difficulties, and overall risk is reduced.

f) **Risk** is measured by the degree of uncertainty in the quantitative estimates established for resources, cost, and schedule. If project scope is poorly understood or project requirements are subject to change, uncertainty and risk become dangerously high.

The software planner should demand completeness of function, performance, and interface definitions (contained in a *System Specification)*. The planner, and more important, the customer should recognize that variability in software requirements means instability in cost and schedule.

LECTURE # 30

6. ESTIMATION

6.3 Decomposition Techniques

Software project estimation is a form of problem solving, and in most cases, the problem to be solved (i.e., developing a cost and effort estimate for a software project) is too complex to be considered in one piece. For this reason, we decompose the problem, re-characterizing it as a set of smaller (and hopefully, more manageable) problems.

Before an estimate can be made, the project planner must understand the scope of the software to be built and generate an estimate of its "size."

6.3.1 Software Sizing

The accuracy of a software project estimate is predicated on a number of things:

(1) The degree to which the planner has properly estimated the size of the product to be built
(2) The ability to translate the size estimate into human effort, calendar time, and dollars (a function of the availability of reliable software metrics from past projects)
(3) The degree to which the project plan reflects the abilities of the software team
(4) The stability of product requirements and the environment that supports the software engineering effort.

As project estimate is only as good as the estimate of the size of the work to be accomplished, sizing represents the project planner's first major challenge.

In the context of project planning, size refers to a quantifiable outcome of the software project. If a direct approach is taken, size can be measured in LOC. If an indirect approach is chosen, size is represented as FP.

There are four different approaches to the sizing problem:

1. **"Fuzzy logic" sizing:** This approach uses the approximate reasoning techniques that are the cornerstone of fuzzy logic. To apply this approach, the planner must identify the type of application, establish its magnitude on a qualitative scale, and then refine the magnitude within the original range. Although personal experience can be used, the planner should also have

access to a historical database of projects8 so that estimates can be com- pared to actual experience.

2. **Function point sizing**: The planner develops estimates of the information domain. Its characteristics will be discussed later in the session.

3. **Standard component sizing:** Software is composed of a number of different "standard components" that are generic to a particular application area. For example, the standard components for an information system are subsystems, modules, screens, reports, interactive programs, batch programs, files, LOC, and object-level instructions. The project planner estimates the number of occurrences of each standard component and then uses historical project data to determine the delivered size per standard component. To illustrate, consider an information systems application. The planner estimates that 18 reports will be generated. Historical data indicates that 967 lines of COBOL [PUT92) are required *per* report. This enables the planner to estimate that 17,000 LOC will be required for the reports component. Similar estimates and computation are made for other standard components, and a combined size value (adjusted statistically) results.

4. **Change sizing:** This approach is used when a project encompasses the use of existing software that must be modified in some way as part of a project. The planner estimates the number and type (e.g., reuse, adding code, changing code, and deleting code) of modifications that must be accomplished. Using an "effort ratio" [PUT92) for each type of change, the size of the change may be estimated.

6.4 Estimation – Tools

i. Work Breakdown Structure (WBS)

a) Dividing into Logical Units/Tasks

To correctly estimate the effort, size, or cost of completing a project, it is important to be aware of the multiple tasks that comprise a project. You can divide a project into logical units or tasks by using the WBS technique.

Creating a WBS is a prerequisite for any estimation activity. It enables you to conceptualize an abstract entity, such as a project, into distinct, independent units.

After dividing a project into tasks, you can categorize them as logical, broad tasks. For example, tasks, such as drawing up a marketing strategy, planning a phase-wise product release, and interacting with media agencies and the production department can be compiled under a common category, Marketing.

b) Benefits of Using a WBS

Using a WBS provides a number of benefits to the management and to the development teams.

First, it gives the management an idea about the size and complexity of the project.

Second, it helps in planning, scheduling, and monitoring a project realistically. This is possible because all the tasks in the project can be preformed measurable targets for each task.

To aid planning, scheduling, and monitoring a project, you can use tools such as:

- Program Evaluation and Review Techniques (PERT)
- Critical Path Method (CPM)
- Timeline charts
- Gantt charts

These tools use WBS as the fundamental basis for assessing resources to tasks, computing the number of days needed, and the cost required to complete the tasks.

ii. Measuring Effort for a Project

Measuring the effort for a project is a specialized activity. It enables you to derive cost estimates that are critical for project management. An incorrect measurement of effort at the beginning of project can result in inaccurate project plans and frequent slippages. It can also lead to inaccurate cost estimates, which can cause steep cost deviations between estimated and actual cost values.

There are many techniques that you can use to accurately estimate effort, such as:

a) Source lines of Code (SLOC)
b) Function Point (FP)
c) Constructive Cost Model (COCOMO)

These are quantitative estimation techniques because they rely on the use of formulae to calculate effort.

In addition to quantitative estimation techniques, there is a human-based technique known as:

d) Delphi technique

This technique is based on soft skills and relies more on human factors, such as collecting information during group discussions.

a) SLOC- Technique

At the beginning of a software project, it is important to determine the size of the project. The project size helps determine the resources, effort, and duration of the project.

There are many techniques to calculate the size of a software project. You can calculate the size by using a directly measurable technique, the SLOC technique.

It is defined as the source lines of code that are delivered as part of the product. The effort spent on creating the source lines of code is expressed in relation to thousand lines of code (KLOC).

The SLOC technique is an objective method of estimating the size because there are no multiple ways of calculating the lines of code. Therefore, the effort estimate is close to being accurate.

This technique includes the calculation of lines of code, documentation of pages, inputs, outputs, and components of a software program. Components are again of multiple types, such as reports, screens, and files.

The SLOC technique is also used to directly calculate the effort to be spent on a project.

Following figure is a simple example of counting source lines of code. The code displayed here contains four lines of code.

```
If (emp_code<=1200)
        Print (“Welcome to the Inventory database.”);
Else
        Print (“Access denied to the Inventory databases.”);
```

– Counting SLOC

You can use the SLOC technique to estimate the effort required for a project when the programming language and the technology to be used are predefined. In addition to the programming language and technology, the complexity and effort required to write a program should be easily predictable.

The use of the SLOC technique requires that the technology or language remain unchanged throughout the project. Generally, you can use the SLOC technique when you are using third-generation languages, such as FORTRAN or COBOL.

While counting the source lines of code, there are some general considerations that you need to keep in mind. However, these can vary in every organization.

The general considerations include the following:

- *Only the delivered lines of code are included in SLOC calculation. For example, test drivers and other support software are not part of the number of lines developed for a project.*
- *Only the source lines of code written only by the development team as counted. This excludes the code created by applications generators.*
- *Only declaration statements are counted as source lines of code. This excludes comments inserted to improve the readability of programs.*

– **Disadvantages of Using SLOC**

Despite being accurate in providing figures to calculate the effort required for a project, the SLOC technique has a drawback.

The SLOC technique is language-dependent. *The effort required to calculate source lines of code may not be the same for all languages.*

For example, to conceive and write 8 lines of code that accomplish a task in the assembly language may require 15 minutes. However, you may need only five minutes to complete the same lines of code if it is written in Visual Basic.

b) FP Technique

The FP technique is a direct indicator of the functionality of a software application from the user's perspective. This is the most popular technique used to estimate the size of a software project.

This fact is further supported by a quote of Capers Jones, chairman of Software Productivity Research, Inc. in Burlington, Massachusetts, on page 1 of Computer Finance brought out in November 1997. He quotes *"80% of the Fortune 500, are using function points, at least somewhere in their application development organizations"*.

You use the FP technique to estimate the total size of a project. The total size of a project is estimated as a single FP value. After calculating the total size of a project in FP, you divide the total FP into the different phases of the SDLC. This way, you can determine how much effort per FP is required in that particular phase.

For example, the testing phase is planned for 20 FP of work. The project managers, based on their past project experience, determine the amount of effort in man/person months required in the testing phase.

Similarly, you can express the cost required to complete FP of work for a particular phase. At the end of a project, you can also express the number of defects reported in terms of per FP for a phase.

– **Features of Function Points**

The total size of a software project is expressed in total function points. It is independent of the computer language, development methodology, technology, or capability of the project team developing the software project.

The specific user functionality of the application is evaluated in terms of relation to what is delivered by the application and, not how it is delivered. Only user-requested and user-defined components are counted. To calculate FP for a project, some major components are required.

The major components and their relationships are represented in following figure.

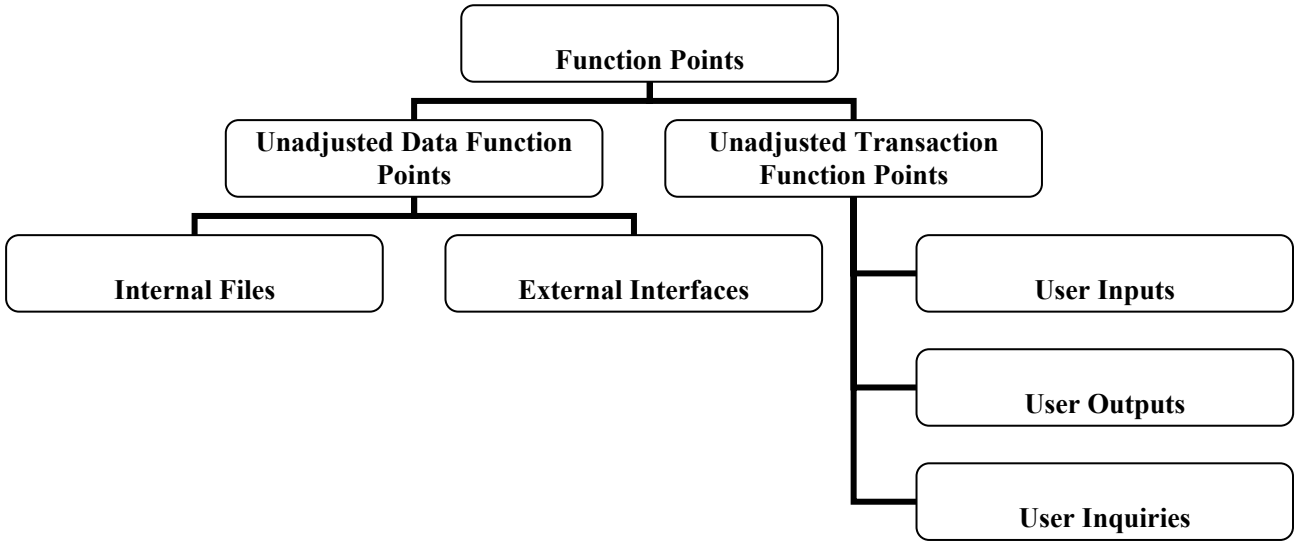


Figure: *Function Points Components and their Relationship*

You can calculate the function point estimates for a project or a particular phase by following four steps:

1. *Identify the unadjusted function points.*
2. *Calculate total GSC s.*
3. *Calculate Value Adjustment Factor (VAF)*
4. *Apply a formula to calculate Adjusted FP (AFP)*

– **Advantages of Using Function Points**

Function points are language-and technology-independent. Therefore, you can use them to estimate any kind of project. They can also be used to estimate the effort, cost, and schedules of projects that use the Prototyping and Spiral models because such projects have uncertain user and project requirements.

In addition, you can use function points as a project estimation technique when you anticipate changes in the middle of a project. These changes may disturb the estimates if, you had used SLOC to estimate the effort, cost, or size of a project.

The FP estimation uses a subjective and holistic approach for project estimation. Consequently, the estimates calculated by using the FP are unlikely to be incorrect.

– **Disadvantages of Using Function Points**

Estimation by using FP generally uses data from past projects for assigning weights to GSC s and the information domain values. To be able to do this realistically, it is important for the organization to have developed similar projects in the past.

The organization should also be prepared with adequate data and tools for FP estimation of the new project.

In addition, FP provides a vague estimation. This characteristic of FP does not usually provide precise or approximate estimates of the effort, cost and size of a project.

Consequently, at the end of a project, deviations from the estimated to the actual, values of each of the factors maybe quite extreme.

LECTURE # 31

6. ESTIMATION

6.5 Estimation – Tools

iii. Measuring effort for a project

c) Constructive Cost Model (COCOMO)

The COCOMO technique is another popular estimation technique. Dr: Barry Boehm propounded this technique in 1981. COCOMO uses cost driver attributes to calculate the effort and duration of a project. The COCOMO technique has three levels of implementation. With each level, the complexity of the model increases. The levels of the COCOMO technique are:

i. Basic
ii. Intermediate
iii. Advanced

i. Basic COCOMO

The basic COCOMO technique estimates the effort and cost of a software project by using only the lines of code. You, use basic COCOMO when you need a rough estimate of effort, such as during maintenance projects. This is because in such projects, a majority of the work is already completed. Estimating the effort in the basic COCOMO technique involves three steps.

1. ***Estimating the total delivered lines of code***
2. ***Determining the effort constants based on the type of the project***
3. ***Substituting values for lines of code and effort constant in a formula***

You have already seen how the total delivered lines of code are estimated. The next step in the COCOMO model is to determine the type of the project being developed. The basic COCOMO technique considers three types of projects to calculate effort.

- ***Organic***
- ***Embedded***
- ***Semidetached***

Organic projects have sufficient and defined objectives. The organizations that undertake organic projects have ample experience in development and use small development teams. These are simple business and financial applications; such as a banking system and inventory system.

Embedded projects have stringent and specialized hardware, software, and human resources requirements. Organizations usually have less experience in developing such projects. Examples of such projects include real-time operating systems (RTOS), industrial automation systems, and sophisticated space and aviation systems.

Semidetached projects are a combination of the preceding two types of software projects. A new operating system and a database management system (DBMS) are examples of such projects.

The last step in calculating effort by using the COCOMO technique is to substitute the values of lines of code and effort constants in the following formula:

$E_j = a1 * (KLOC)^{a2}$

In the formula, **Ei** is the effort for a project. The effort constants, **al** and **a2** depend on the type of project being developed.

ii. Intermediate COCOMO

Calculation of effort by using the intermediate COCOMO technique involves an additional step of calculating the effort adjustment factor (EAF). The effort adjustment factor is calculated by assigning ratings to 15 cost driver attributes.

These cost driver attributes relate to the various aspects of a software project, such as project, product, personnel, and computer attributes. Using the intermediate COCOMO technique, you can accurately estimate effort and cost required for a project. Accurate estimates are very helpful to start new development projects.

Calculating the effort by using the intermediate COCOMO technique is a three-step process:

1. Estimate the initial development effort by using SLOC. To do this, you use the following formula:
   $E_i = a1 * (KLOC)^{a2}$

In the formula the initial development effort, KLOC refers to 1,000 lines of code. The constant values a1 and a2 differ with every project.

2. The second step is to determine the relevant cost driver attributes that affect your project intensively. This provides you with the value for EAF.

Table 1 summarizes 15 commonly used projects, personnel, and product-related cost driver attributes. The values for each cost driver under each rating are filled in by an organization based on past experience.

Table 1: *Cost Driver Attributes*

Cost Drivers	Rating					
	Negligible	Low	Average	High	Very high	Extremely critical
Analyst Capability (ACAP)						
Programmer Capability (PCAP)						
Programming Language Experience (LEXP)						
Virtual Machine Experience(VEXP)						
Required Software Reliability (RELY)						
Database Size (DATA)						
Software Product Complexity (CPLX)						
Execution Time Constraint (TIME)						
Main Storage Constraint (STOR)						
Computer Turnaround Time (TURN)						
Virtual Machine Volatility (VIRT)						
Use of Software Tools (TOOL)						
Modern Programming Practices (MODP)						
Required Development Schedule (SCED)						

3. Finally, you calculate the actual effort by multiplying the weighted cost driver attributes with the initial effort estimate. Typically, the val1,les that rate each cost driver attribute range from 0.9 through 1.4. For example, if software reliability (RELY) is of prime importance according to the requirements specifications, it is provided a rating of high or a value of 1.4. Similarly, if the time to execute a software program is of negligible importance, you assign a rating of low or a value of 0.9. For software attributes that are of mediocre importance, you can assign a value between 0.9 and 1.4.

Usually, in organizations, the average rating is assigned a static value of 1.0. To calculate the estimated effort using the intermediate COCOMO technique, you use the formula:

E = EAF * Ei

Consider an example for using the intermediate COCOMO technique to calculate the estimated total effort of a project life cycle. In a customized insurance project, there are four modules. The total effort estimate of the modules is 3.0 KLOC. The management has identified four cost driver attributes with the respective multiplying factors that might affect the project most. In this situation, the values of al and a2 are 3.2 and 1.05, respectively, because the insurance project is an organic project. Therefore, you apply the following formula to calculate the initial effort estimate.

$Ei = a1 (KLOC)^{a2}$
Ei =3.2 * 31.05
Ei = 3.2 * 3.16
Ei = 10.11

The values assigned to the cost driver attributes that are applicable to a particular software application are displayed in Table 2. According to the table, the time to execute a software program is of high importance. Therefore, the attribute TIME is assigned a value of 1.35. In contrast, the software application does not require a very high analyst involvement. Therefore, the value assigned to ACAP is very low or 0.95. Using the same logic, the values for other cost driver attributes are assigned.

***Table* 2:** *Applicable Cost Driver Attributes*

Applicable cost driver attributes	Rating	Multiplying factors
CPLX	High	1.2
TIME	Very high	1.35
ACAP	Low	.95
MODP	Average	1.00

Referring to Table 7.9, EAP can be calculated as,
EAF = 1.2 * 1.35 * 0.95 * 1.0 EAF = 1.53

After obtaining the values for the variables Ei and EAF, you can substitute these values in the formula to calculate the total effort.

E = EAF * Ei
E = 1.53 * 10.11

E = 15.5 person months

iii. Advanced COCOMO

The advanced COCOl\l10 technique uses the steps of the intermediate COCOMO technique. In addition, it uses costs driver attributes assigned to each phase of the SDLC such as analysis and design.

- **Applicability of COCOMO**

COCOMO is flexible and capable of using SLOC, FP, and even object points. Object points are measurable code sections in an object-oriented programming language, such as C++, Ada, and Java.

You can use COCOMO when the size of a project is extensive and the requirements of the project are vague. In contrast, SLOC and FP can be used *for* projects where either the requirements are more or less known or developers possess the relevant experience in developing projects.

COCOMO is suitable for complex and sophisticated projects that are expected to operate within intensive hardware, software, and personnel constraints.

Generally, you can use COCOMO when the software development environment is new to an organization. In addition, you can use COCOMO when you do not have baseline data about past projects. However, you need complete data about your current project to assign weight age to each cost driver attribute. You can use FP or SLOC techniques when you have enough past project data to assign accurate weight age to the 14 GSC s and the various information domain value elements.

d) Delphi Technique

The Delphi technique is a Human-based estimation technique. Human-based estimation techniques use human experience and analytical skills to estimate the size, productivity, and effort required for a project. This is a trusted technique and is widely used in many established organizations to facilitate practical and reasonable estimation.

The rationale of using the Delphi technique is that when many experts independently arrive at the same estimate on the basis of similar assumptions, the estimate is likely to be correct.

The Delphi technique has eight basic steps:

1. Identify the terms that need to perform the estimation activity. In an estimation activity meeting, three distinct groups of people need to be present.

- *Estimation experts:* They usually consist of groups of five or six experienced project managers. The estimation values provided by the project managers are based on past project history and their knowledge. However, only those project managers should b~ invited for estimation whose experience of a past project matches that of the current project. Otherwise, estimation values may turn out to be far from realistic.
- *Estimation coordinator:* An estimation coordinator is very similar to a moderator in a usual meeting. The coordinator facilitates the meeting and ensures that the goals of the meeting are fully achieved.
- *Author:* An author is similar to a recorder of minutes in a meeting.

2. The author presents the project details including clients' needs and system requirements to the group of experts. The author also describes the expectations from the group. The author and experts jointly identify the tasks that need to be estimated. They also identify the valid assumptions that they need to consider while estimating. For example, while estimating the effort needed to create a high-level design, they can assume that the SRS document is approved by the client.

3. The author and experts arrive at a consensus that any estimation with a specific variance value will not be accepted. For example, they may decide that any variance above 25 percent will not be accepted as an estimation value for computing the project effort or the productivity.

4. The coordinator prepares a list of tasks jointly decided by the team and distributes the list to all experts. These tasks comprise a project plan.

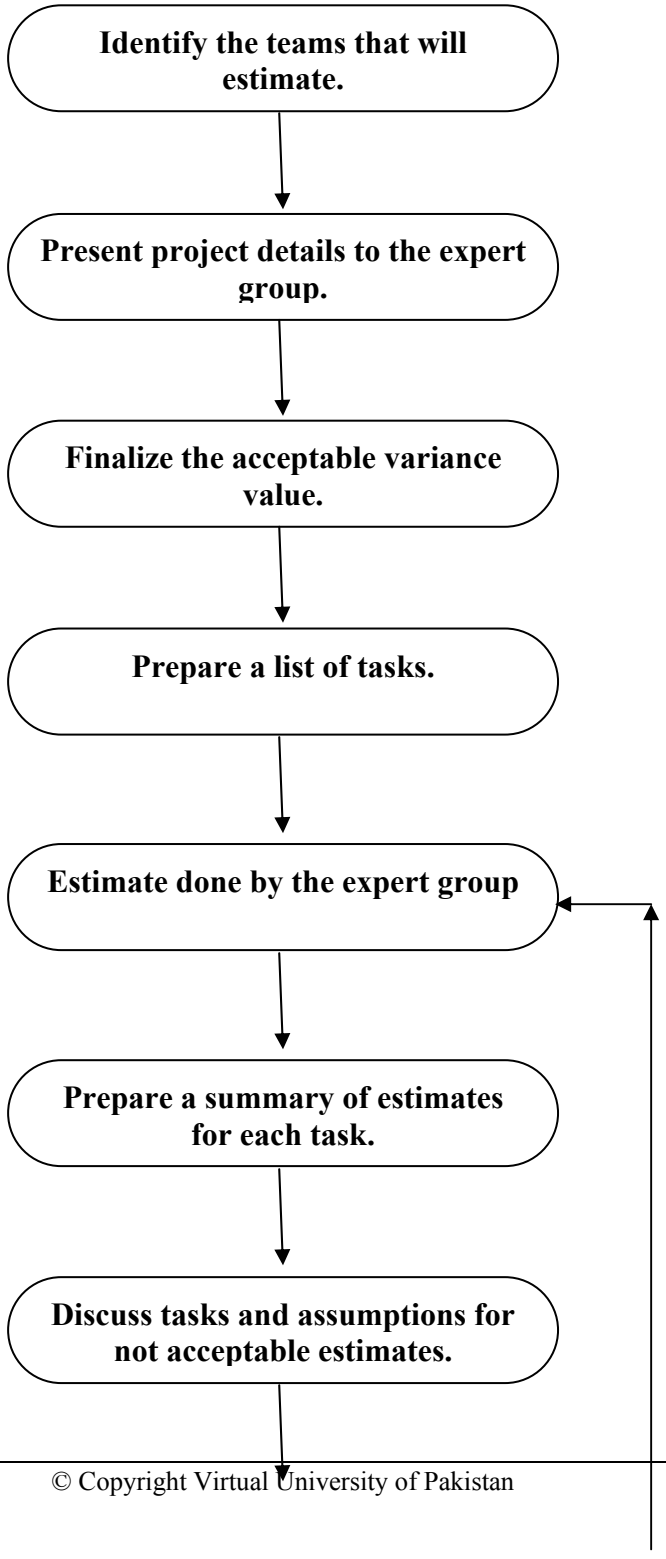
5. The experts independently make their estimates for each task. After recording their estimates, they hand over their estimates to the coordinator. This is a critical step. While making estimates, no discussions or consultations are permitted because a mutual discussion may influence the estimation logic of the fellow experts. The coordinator and the author jointly ensure this.

6. The coordinator prepares a summary of estimates for each task in a table as represented in Table 7.10. After calculating the percentage of variance, the coordinator marks each task as accepted or not accepted based on the agreed accepted value.

7. The coordinator hands over the summary to the group of experts and the author. The group of experts and the author discuss tasks and assumptions where the percentage of variance is more than the acceptable level. The maximum and minimum estimates of tasks are not disclosed or discussed. For example, in Table 7.10, the group and the coordinator do not accept the high-level design task because it exceeds the agreed variance value of 25%. Therefore, the team would discuss this task to estimate its maximum and minimum effort afresh. To resolve

the high percentage of the variance value, some tasks may be broken down further or combined. This activity of breaking down tasks into smaller levels involves fresh estimates for those tasks at the smaller levels.

8. Revert to step 5 and repeat the steps. You do this until all tasks are assigned estimates that have an acceptable percentage of variance value. Figure 7.4. summarizes the steps of the Delphi technique in the form of a flowchart.



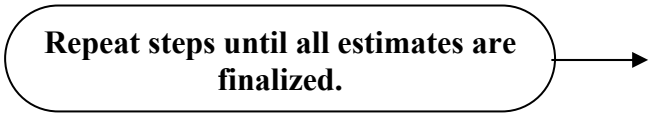


Figure 1: *Steps of the Delphi Technique*

The Delphi technique is a simple and subjective method of estimation. However, it is a very effective method because most of the estimates are tried and tested. You can use this method if the project is small or if you have the data and expertise that can enable unambiguous estimates.

LECTURE # 32

7. Work Breakdown Structure

7.1 WBS Preview

i. Salient features

Large, complex projects are organized and comprehended by breaking them into progressively smaller pieces until they are a collection of defined "work packages" that may include a number of tasks.

A $1,000,000,000 project is simply a lot of $50,000 projects joined together.

Work breakdown structure (WBS) is a technique to decompose the project for the purpose of management and control. It provides the framework for organizing and managing the work.

A project may be divided into subprojects. Subprojects are subdivided into smaller, more manageable work components (work packages) at the lowest level.

The work packages may be further decomposed to project activities. It provides the framework for organizing and managing the work.

A project may be divided into subprojects. Subprojects are subdivided into smaller, more manageable work components (work packages) at the lowest level. The work packages may be further decomposed to project activities.

The WBS is commonly used at the beginning of a project for defining project scope, organizing Gantt schedules and estimating costs.

It lives on, throughout the project, in the project schedule and often is the main path for reporting project costs.

WBS includes activities like management, procurement, installation, software development etc.

Many of the WBS development tasks are derived from the development method that will be used, and from the design and architecture of the system.

WBS is related to planning and scheduling a project

It is a functional decomposition of the tasks of the project.

It starts with the end objective required and successively subdividing it into manageable components in terms of size and complexity of:

- ***Program,***
- ***Project,***
- ***System,***
- ***Subsystem,***
- ***Components,***
- ***tasks, subtasks, and***
- ***work elements***

The WBS is first and foremost a technical data gathering structure, developed so that the achievement in technical progress can be measured and analyzed against a formal baseline plan.

The WBS aids the customer in understanding the status of the project as time elapses. The WBS aids the customer's customer in understanding the status of the project.

All managers, internal and external need to use the planning and status information within the WBS structure to:

- ***Aid in the adjustment to the current program paths and***
- ***For maximizing the attainment of short term and long term goals***

ii. WBS-What is it?

WBS is a definition of a project in terms of its work or a deliverable-oriented grouping of project elements that organizes and defines the total scope of the project.

Each descending level represents an increasingly detailed definition of the project's work.

It's an outline of the work of the project, not the work itself, created by those doing the work – that may include all functional stakeholders.

iii. WBS- What it contains?

- Maps all contractual obligations (SOW) regarding deliverables
- Details project objectives
- Detailed enough to meet performance (measurable) objectives
- Contains built-in WBS and Project Plan review and update

iv. WBS- When is it produced?

The WBS is produced following the development of the scope statement, before the schedule. It is a "bridging" document between the Scope and Schedule. (what and when)

v. Uses of WBS

- Defines 100% of the scope and can communicate the scope of the project without the presence of the scope statement or document.
- Communicates effectively to all stakeholders
- Defines and clarifies
- Boundaries (Life cycle of the project – not the product)
- Deliverables
- Refines Scope

Defines scope in project management language (perhaps down to work package level which may lead to development of project schedule)

It's not a single document that can be mistaken for the project plan, schedule or scope statement

7.2 WBS- A Mandatory Management Tool

i. WBS as a Project Management Tool

A WBS is a valuable management tool that is used throughout all life-cycle phases to:

a. Manage Risk

It helps in managing risk by providing insight into technical aspects of program management.

b. Manage Costs

A WBS can be used to help make program management decisions.
For example, if the costs of an element in the WBS are too high, the WBS can be used to identify possible tradeoffs. Identifying and analyzing tradeoffs can help the manager decide how best to stay within budget.

c. Assign Work

The WBS is also useful for determining an acquisition strategy and/or assigning work. The information contained in the WBS can help a Program Manager

develop a statement of work that describes what products or services are to be delivered.

d. Schedule and Track activities

A schedule of key events can be developed for each element in the WBS. Completion of these key events is then tracked.

- Schedule and Track activities
- The work breakdown structure defines all tasks to be performed during the development of the project. This will include tasks from such project categories as:

  - ***Software development***
  - ***Installation***
  - ***Maintenance***
  - ***Management***
  - ***Training***
  - ***Procurement***
  - ***Documentation***

e. Align with Terms of Reference and scope of project

- At all times, any, work being performed by a member of the software project team must be part of a WBS task: No member of the team should ever perform any task that does not appear in the WBS list of tasks.
- The WBS is essentially a management tool that provides the ability to assign well-define tasks to members of the development team.
- It is through the WBS that progress is monitored as tasks are completed and potential problems are discovered.
- New tasks that were overlooked are identified, and estimates are revised based on the actual resources used for completed tasks.

f. Report Expense

- The WBS is also a budgetary tool that provides a means of charging each development activity to the appropriate section in the project budget.

- This is one of the basic methods for planning and monitoring project expenditure.

- There are mainly computerized utilities available to support the maintenance of the WBS.

- These utilities run both on small PC type computers and large mainframes.

- WBS utilities are often available as part of a manager's general planning utility, and provide other scheduling and monitoring features, such as PERT analysis and report generation.

LECTURE # 33

7. Work Breakdown Structure

7.3 WBS- A Mandatory Management Tool

i. The WBS as a project management tool

g) Mapping WBS for Cost Management

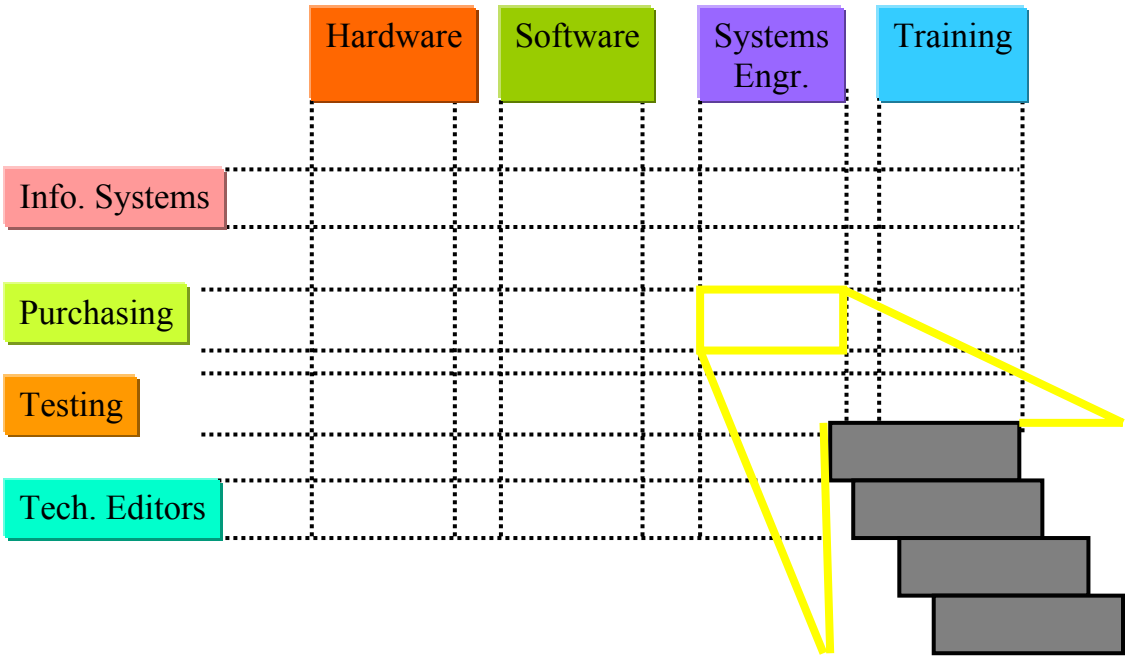
In a product-oriented WBS, functional categories of work may form "cost accounts" within a WBS element. Cost account managers are responsible for a functional area's contribution to a WBS element. Cost accounts from several departments or functions may combine into one WBS element.

Internal department planning for a cost account will be made up of individual work packages. A work package will typically have its own budget and schedule. Work packages should be small enough to be executed by individuals or small groups in a single department, and they should be of relatively short schedule duration. A small project might define a maximum work package size as two weeks of effort. Larger projects will assemble larger work packages that can be appropriately managed and controlled.

The project manager will have to decide to what degree employment of various details of WBS implementation will benefit the efficient management of the project. On a very small project, a formal WBS may serve no useful purpose, but it can become valuable if project size or complexity start to increase.

As an organization's project management environment matures, or as larger size and complexity are encountered, application of the WBS concept can evolve from an *ad hoc* list of tasks, to time-phased activity lists, task lists clustered by project deliverables and services, or an end-product focused WBS fed by cost accounts and work packages.

If you are using MS-Project or a similar project management software application, you may encounter the WBS as a vertical list with indents to show structure. This will be compatible with the Gantt View data entry screens. While some software packages provide a separate WBS view, you could prepare your WBS in the vertical format using a word processor, and then cut and paste your WBS into your project management software package.



WORD OF CAUTION

- The WBS is a useful tool provided it is constantly updated.
- It should be updated periodically together with the project development plan and the project schedule.
- It is reasonable to expect the WBS list to have tasks added, modified or even removed as project development progresses.

LECTURE # 34

7. Work Breakdown Structure

7.4 WBS- A Mandatory Management Tool

h) Characteristics of a High-Quality WBS

a) Review and signoff from top to bottom

b) Includes logical flow and is hierarchical in nature

c) Clear and concise

d) Provides ability to roll-up information to higher levels

e) Receives 100% team buy-in

f) Should have at least 2 levels: Level 1 defines 100% of the service/product/result; Level 2 defines the deliverables in terms of work (groupings)

g) Project Management (and sub-contract management) at Level 2.

h) The deliverables in the WBS must match the scope or contract (WBS should not contain work that is not defined in the scope – Scope should not describe work not contained in the WBS)

i) All deliverables should be accounted for regardless of responsibility

j) Every WBS element should be clearly defined – or should be clarified in the WBS Dictionary

k) Features

- The WBS should contains 100% of the work defined by the scope or contract
- Development of WBS should involve the entire project team
- Should be deliverable-oriented
- Should captures all deliverables (Internal, External, Interim) in terms of work to be completed

l) Usefulness

- Should define the context of the project and clarifies the work
- Should communicates project scope to all stakeholders in terms of the work to be completed
- Is "in sync" with the scope statement and project schedule
- Implies and allows for continual improvement/update of the WBS to maintain current-ness and "vitality" within the project

m) WBS Is Not

- A single document that substitutes for the project schedule or project plan
- The project schedule
- A listing of tasks or activities

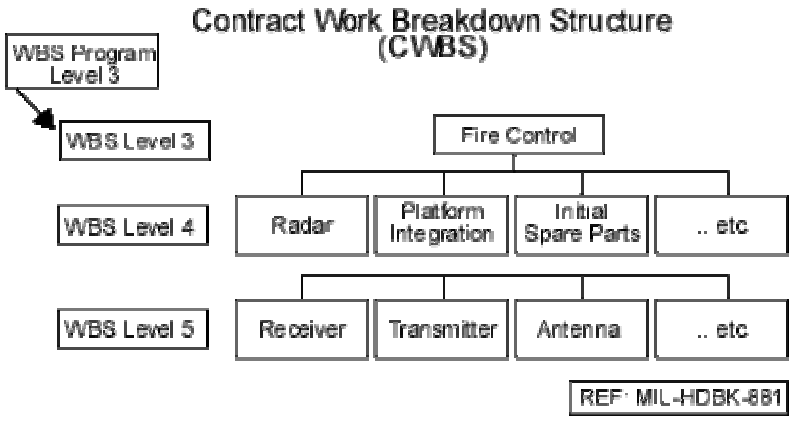
i) **Types of WBS**

The two types of Work Breakdown Structures are:

a) Program Work Breakdown Structures
b) Contract Work Breakdown Structures

a) **Contract WBS**

A Contract WBS extends the Program WBS to a lower level in order to provide management and cost information to the Government. It includes all the elements for products (e.g., hardware, software, data, or services) that are the responsibility of the contractor. It must be consistent with the Program WBS.



– **Contract WBS: Contractor's Responsibility**

Contractors may extend the work breakdown structure to whatever level they feel is necessary to manage the program. Contractors also use the Contract WBS to define work packages. Work packages are:

Discrete portions of the project that can be charged to a single organization. Building blocks for program management and cost reporting.

Remember that a Contract WBS:

- Is prepared and maintained by each contractor working on different portions of the program.
- Includes the number of levels thought sufficient by the contractor to manage the program.
- Must be updated if changes are made to the Program WBS.

b) **Program WBS**

A Program WBS is defined as "the work breakdown that covers the acquisition of a specific defense materiel item and is related to contractual effort."

A Program WBS is:

- Tailored to each specific program.
- Prepared and maintained by the Government.
- Provides a basis for developing the Contract WBS.

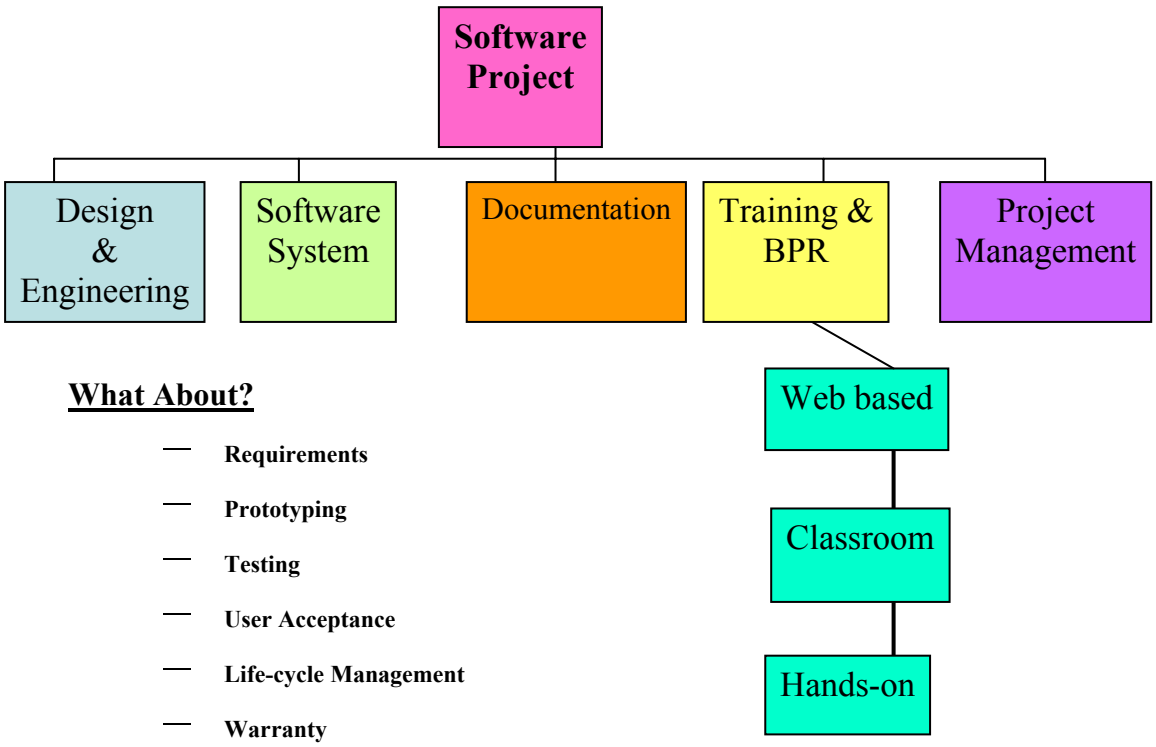
– **Program WBS Levels**

Typically, a Program WBS consists of the upper three levels.

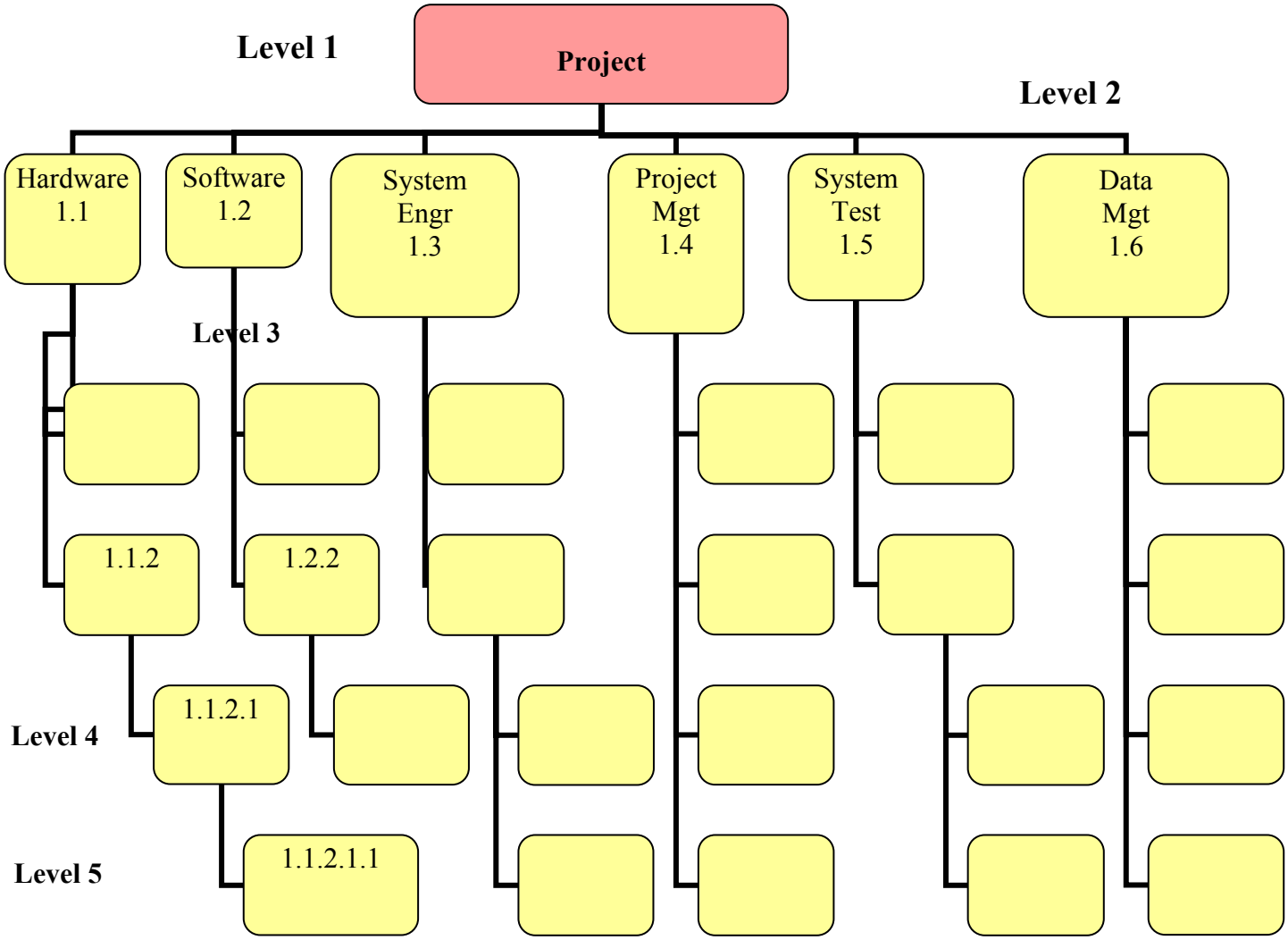
WBS Level 1:	The entire defense materiel item.
WBS Level 2:	Major elements of the defense materiel item, which are subordinate to Level 1.
WBS Level 3:	Elements subordinate to Level 2 elements.

j) Sample WBS

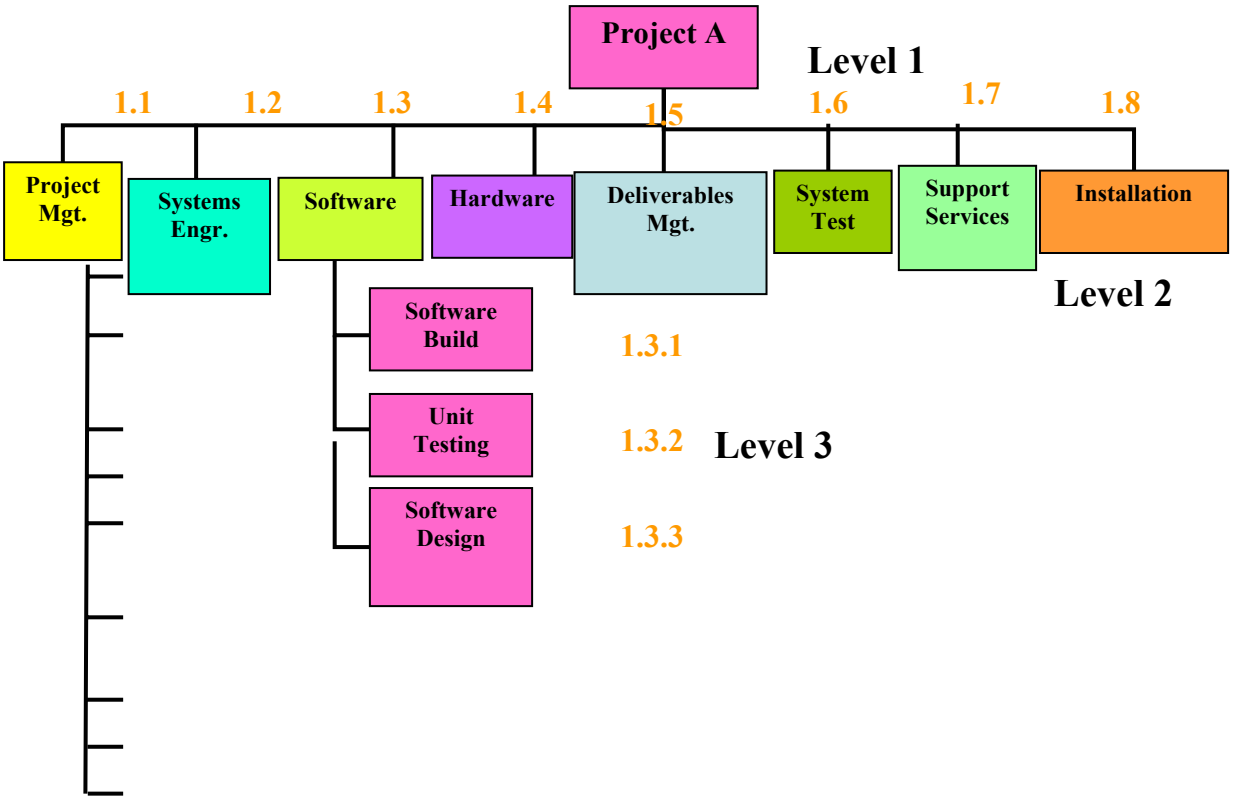
Sample 1: A Product WBS



WBS Sample 2: **System Development Projects**



WBS Sample 3: **Software/Hardware System Development**



READING REFERENCE:

For detailed reading refer to:

(i) Chapter No. 6 entitled "How to handle large projects: Divide and Conquer" of "Software Project management – A Practitioner Approach by E. M. BENNATAN.

(ii) Chapter 5 entitled Software Project Planning" of: SE–A Practitioner Approach by Roger S. Pressman.

(iii) Chapter 7 entitled "Software Project Estimation: Tools and Techniques by NIIT

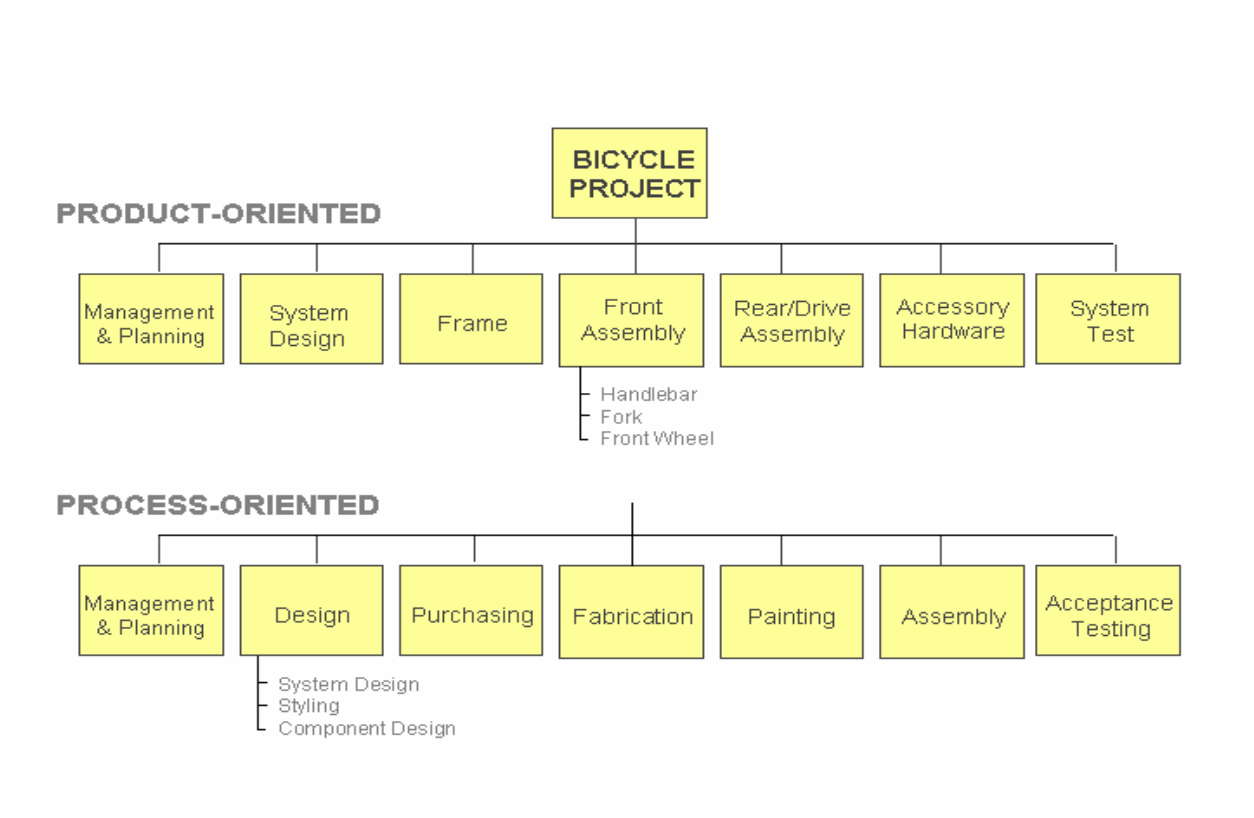
LECTURE # 35

7. Work Breakdown Structure

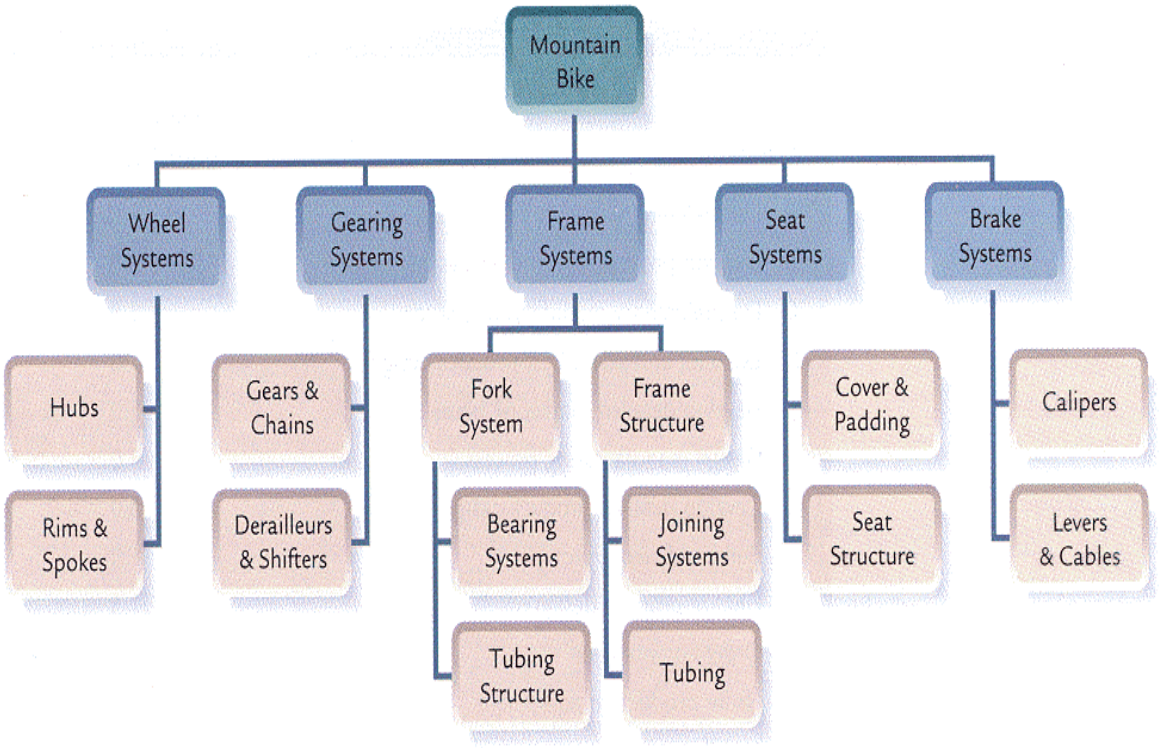
7.5 WBS- A Mandatory Management Tool

iv. Sample WBS

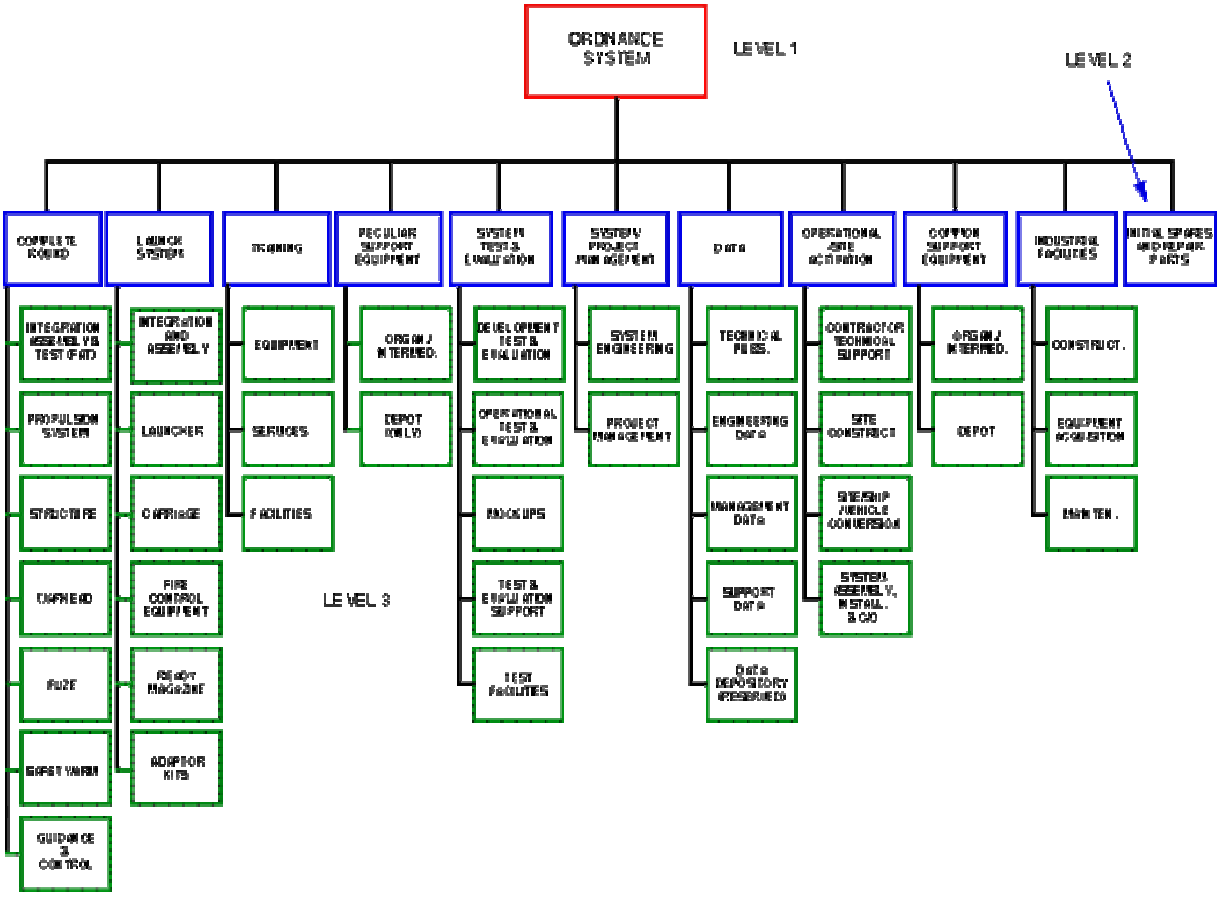
Sample 1: These two graphics illustrate approaching the WBS as a parts list or as a process list



Sample 2: WBS of Manufacturing of a Mountain Bike



Sample 3: This illustrates an example of Ordnance System WBS



The list of activities is often called a Work Breakdown Structure (WBS). The goal is to integrate the WBS, the schedule, and the budget into a written plan.

The WBS reflects activities associated with overall project management, requirements, design, implementation, transition management, testing, training, installation, and maintenance. The project manager is responsible for defining all top level tasks associated with a project and then further decomposing them as planning continues.

An activities list is typically shown in one of two ways. It can be shown as an outline or it can be graphically presented. Two samples of an activities list (WBS) are shown below.

1.0 MANAGEMENT

1.1 Plan Project
1.1.1 Develop Project Plan
1.1.2 Update Project Plan
1.2 Track Project
1.2.1 Prepare status reports
1.2.2 Collect/analyze project metrics
1.3 Perform Quality Activities
1.3.1 Prepare QA Plan
1.3.2 Conduct Reviews
1.3.3 Conduct Audits
1.4 Perform Configuration Management
1.4.1 Prepare CM Plan
1.4.2 Develop Project Library
1.4.3 Manage Change Board
1.4.4 Maintain Configuration Items

2.0 DESIGN

2.1 Prepare Preliminary Design
2.1.1 Develop Enterprise Architecture
2.1.2 Prepare Data Flow Diagrams
2.1.3 Prepare Logical Data Model
2.2 Prepare Detailed Design
2.2.1 Prepare Physical Data Model
2.2.2 Prepare Data Dictionary
2.3 Document Design
2.3.1 Develop Design Specification
2.4 Review Design

3.0 DEVELOPMENT/INTEGRATION

3.1 Develop Software
3.1.1 Develop Server Application
3.1.2 Develop User Interface
3.1.3 Develop XYZ Interface
3.2 Procure Hardware
3.2.1 Procure Server
3.2.2 Procure Workstations
3.3 Procure Software Packages
3.3.1 Procure Database
3.3.2 Procure User Interface Building Tool
3.3.3 Procure Operating System
3.4 Perform Integration Testing
3.5 Convert Data
3.5.1 Develop Conversion Plan
3.6 Develop User Manual
3.7 Transition Management

4.0 ACCEPTANCE TESTING

4.1 Plan Acceptance Test
4.2 Conduct Acceptance Test
4.3 Develop Test Report

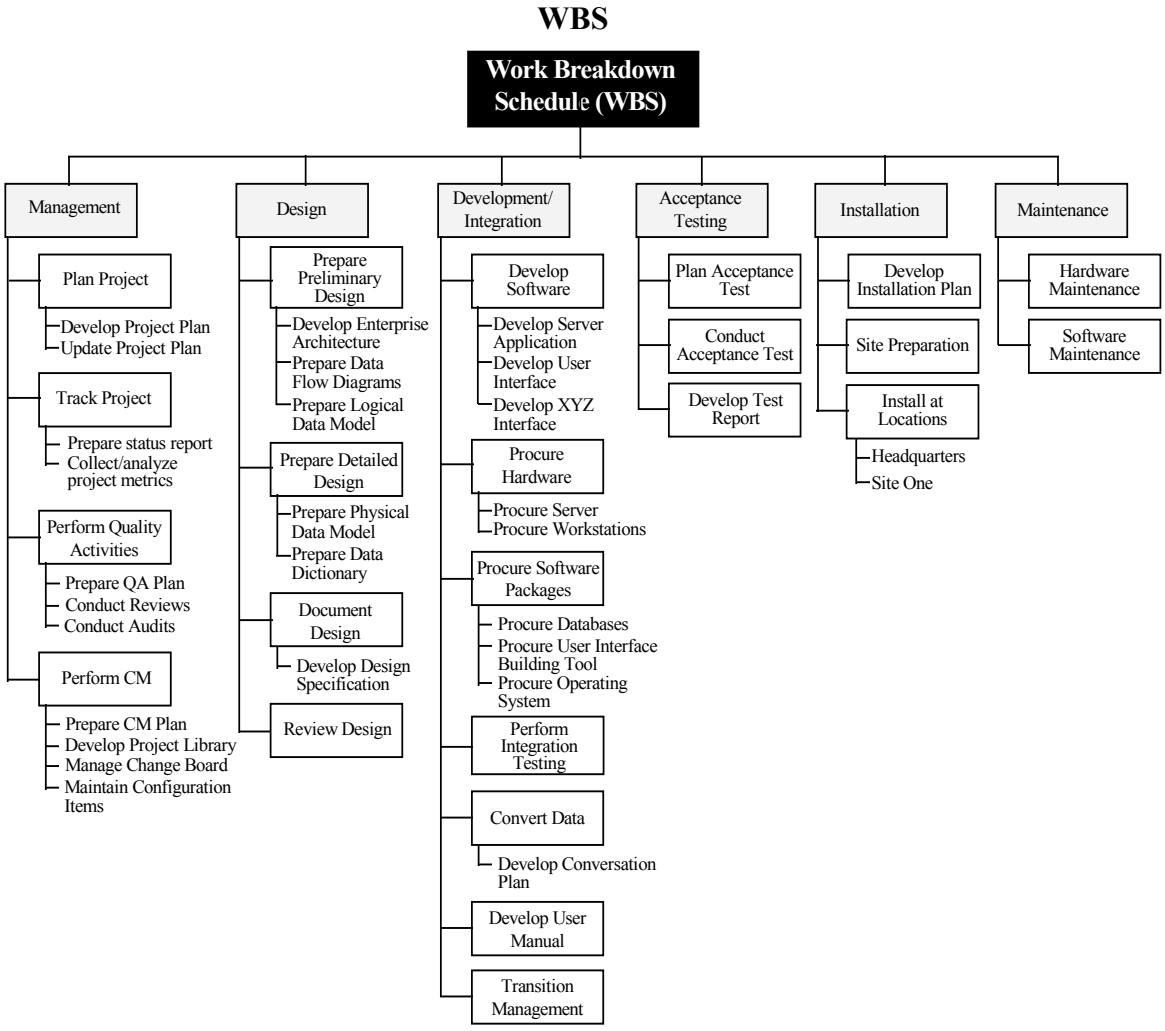
5.0 INSTALLATION

5.1 Develop Installation Plan
5.2 Site Preparation
5.3 Install at Locations
5.3.1 Headquarters
5.3.2 Site 1

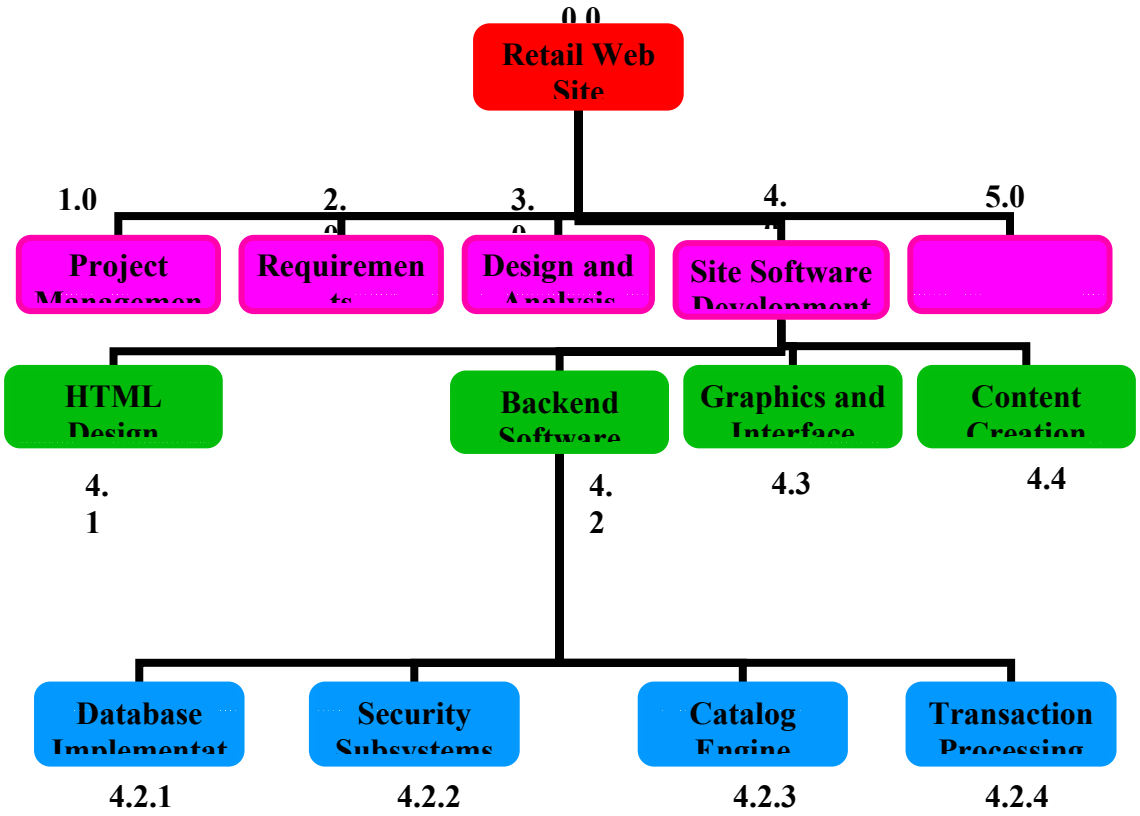
6.0 MAINTENANCE

6.1 Hardware Maintenance
6.2 Software Maintenance

Sample 4: Sometimes, instead of depicting the WBS as an outline, it is shown graphically, as below:



Sample 4: Chart WBS



Sample WBS - Outline Example-1

0.0 Retail Web Site
1.0 Project Management
2.0 Requirements Gathering
3.0 Analysis & Design
4.0 Site Software Development
- **4.1 HTML Design and Creation**
- **4.2 Backend Software**
  - **4.2.1 Database Implementation**
  - **4.2.2 Middleware Development**
  - **4.2.3 Security Subsystems**
  - **4.2.4 Catalog Engine**
  - **4.2.5 Transaction Processing**
- **4.3 Graphics and Interface**
- **4.4 Content Creation**

5.0 Testing and Production

Sample WBS - Outline Example-2

1.10 MANAGEMENT

1.1 Plan Project
- 1.1.1 Develop Project Plan
- 1.1.2 Update Project Plan

1.2 Track Project
- 1.2.1 Prepare status reports
- 1.2.2 Collect/analyze project metrics

1.3 Perform Quality Activities
- 1.3.1 Prepare QA Plan
- 1.3.2 Conduct Reviews
- 1.3.3 Conduct Audits

1.4 Perform Configuration Management
- 1.4.1 Prepare CM Plan
- 1.4.2 Develop Project Library
- 1.4.3 Manage Change Board
- 1.4.4 Maintain Configuration Items

2.0 DESIGN

2.1 Prepare Preliminary Design
- 2.1.1 Develop Enterprise Architecture
- 2.1.2 Prepare Data Flow Diagrams
- 2.1.3 Prepare Logical Data Model

2.2 Prepare Detailed Design
- 2.2.1 Prepare Physical Data Model
- 2.2.2 Prepare Data Dictionary

2.3 Document Design
- 2.3.1 Develop Design Specification

2.4 Review Design

3.0 DEVELOPMENT/INTEGRATION

3.1 Develop Software
- 3.1.1 Develop Server Application
- 3.1.2 Develop User Interface
- 3.1.3 Develop XYZ Interface

3.2 Procure Hardware
- 3.2.1 Procure Server
- 3.2.2 Procure Workstations

3.3 Procure Software Packages
- 3.3.1 Procure Database
- 3.3.2 Procure User Interface Building Tool
- 3.3.3 Procure Operating System

3.4 Perform Integration Testing

3.5 Convert Data

3.5.1 Develop Conversion Plan
3.6 Develop User Manual
3.7 Transition Management

4.0 ACCEPTANCE TESTING
4.1 Plan Acceptance Test
4.2 Conduct Acceptance Test
4.3 Develop Test Report

5.0 INSTALLATION
5.1 Develop Installation Plan
5.2 Site Preparation
5.3 Install at Locations
5.3.1 Headquarters
5.3.2 Site 1

6.0 MAINTENANCE
6.1 Hardware Maintenance
6.2 Software Maintenance

Sample WBS - Outline Example-3

Step	Description	Estimate
1.	**Preparation**	
1.1.	**Developer orientation**	TBD
1.2.	**Requirements gathering**	
1.2.A.	**Use Cases**	TBD
1.2.B.	**Changes to data model**	TBD
1.2.C.	**Non-functional requirements**	TBD
1.2.D.	**Requirements validation**	TBD
1.3.	**Design**	
1.3.A.	**Software architecture**	TBD
1.3.B.	**User interface design**	TBD
1.4.	**Development tools**	TBD
2.	**Construction**	
2.1.A.	**System implementation**	
2.1.A.1	**Unit tests**	TBD
2.1.A.2	**Production code**	TBD

2.1. A.3.	**Integrate Components (mostly done during component implementation)**	TBD
2.1.B.	**Maintenance documentation**	TBD
2.1.C.	**User documentation**	TBD
3.	**Quality assurance**	
3.1.	**Fixing defects reported by users**	TBD
3.2.	**Testing**	
3.2. D.1.	**Test planning**	TBD
3.2. D.2.	**Test implementation**	TBD
3.2. D.3.	**Test execution**	TBD
3.3.	**Reviews of design and code**	TBD
3.4.	**Fixing defects found by testing and review**	TBD
4.	**Transition**	
4.A.	**Release packaging**	TBD
4.B.	**Documentation for other groups**	TBD
5.	**Reflection**	
5.1.	**Postmortem report**	TBD
	Total fixed costs	**TBD**

- **Tasks shown as blue are fixed costs of the project. These should be estimated soon.**

- **Tasks shown as black are proportional to the size of the requirements. A more detailed work breakdown structure will be produced for these tasks when the requirements are better known.**

- **Tasks shown as red depend on the defect rate in the delivered code. These will be estimated after 12 months experience.**

TIP: Label each step uniquely to show its position in the WBS, e.g., Step 1.1.4.A. Use numbers for steps that you intend to do in sequence, and use letters for steps that you intend to do in parallel. E.g., Step 1.1 comes before Steps 1.2.A and 1.2.B, but those two steps may be done in parallel, and Step 1.3 will be done after all 1.2.* steps have been finished. Don't worry about renumbering if you delete a step.

7.7 WBS- Major Steps

a) Project Decomposition

Many complex objects can be viewed as a collection of numerous simpler objects. An appropriate example would be a complex chemical compound that is formed by various molecules, each of which is formed by combining various atoms. The atom, though itself divisible, can be regarded as the smallest particle of a chemical substance.

In a similar way, a complex project can be divided into simpler components. While the full project may be difficult to manage, each component will be easier to handle. Software projects can be decomposed into smaller components in order to provide better estimates of the amount of work involved, or in order to monitor the activities of the various development teams.

The decomposition of a software project is one of the software project manager's first tasks. However, the method of decomposition may differ, depending on the project manager's actual objective. A functional decomposition of a project may not be the same as design decomposition. A functional decomposition divides the project into its basic components from a user's perspective, while design decomposition divides a project into its basic programming components or modules.

Intuitively, it would not appear reasonable to attempt to identify all project components in a single step. Clearly, an iterative procedure that would gradually provide more detail would be easier to use. Iterative methods of this kind are called stepwise refinement, as the decomposition is further refined in each succeeding step.

Figure 1 presents a general illustration of stepwise refinement. The system is initially divided into three top level components. In turn each top level component is further divided into lower level components, and so forth, until the lowest decomposition level is reached.

In a stepwise decomposition of a project, each component decomposes into the components directly below it, so that each step of the decomposition describes the full system, but at a different level of detail. In Fig.1, components 1, 2 and 3 comprise the complete system. For more detail we take the next decomposition step, and find that components 1.1, 1.2, 2.1, 2.2, 2.3, 3.1 and 3.2 now represent the whole system.

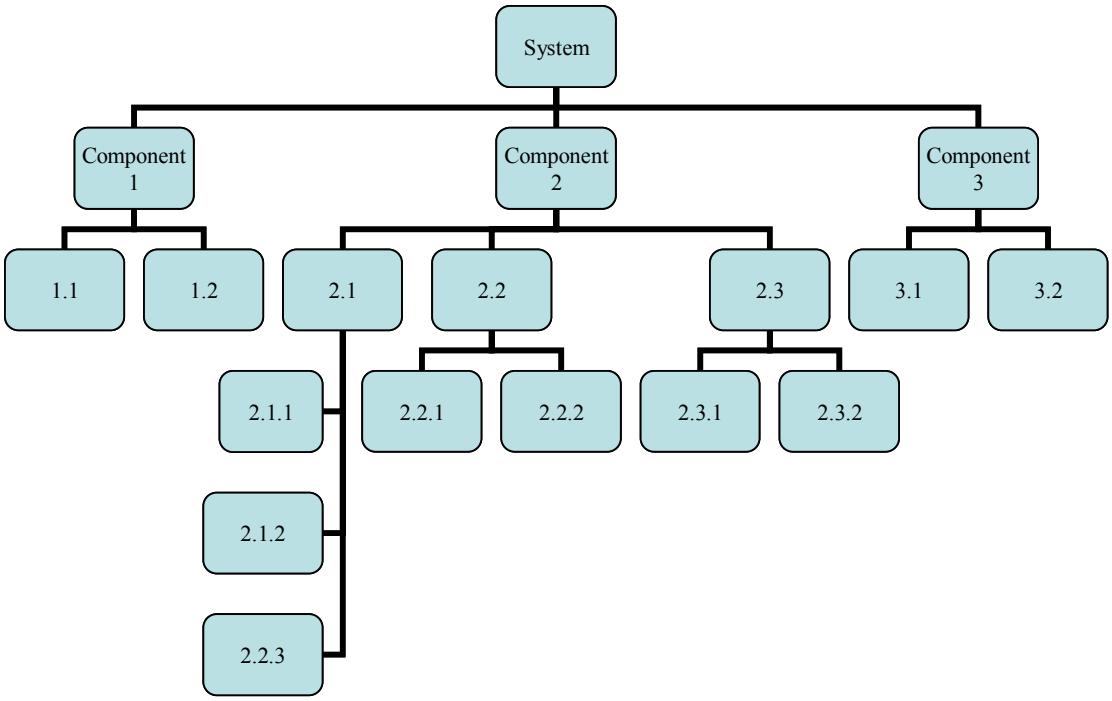


Figure 1: Software decomposition by stepwise refinement

A stepwise refinement diagram looks similar to a hierarchical system chart. However, it is important to understand that stepwise refinement is basically different because the diagram's building blocks are different A hierarchical system diagram describes the hierarchical relationship between components, so that each component in the diagram actually corresponds to a real component in the system. However, in a stepwise refinement diagram, a higher level component is only a name conveniently given to a group of real components that appear below it.

In Fig. 2(a) the Controlled access system software has five low level software components:

Visitor identification, Door lock-control, Access file-manager, Illegal access identification and Alarm activation. Each of these five modules may correspond to an actual software module[1]. The two high level components, Access control and Alarm system, do not exist as actual software modules, and only appear as names given to the two groups of lower level components.

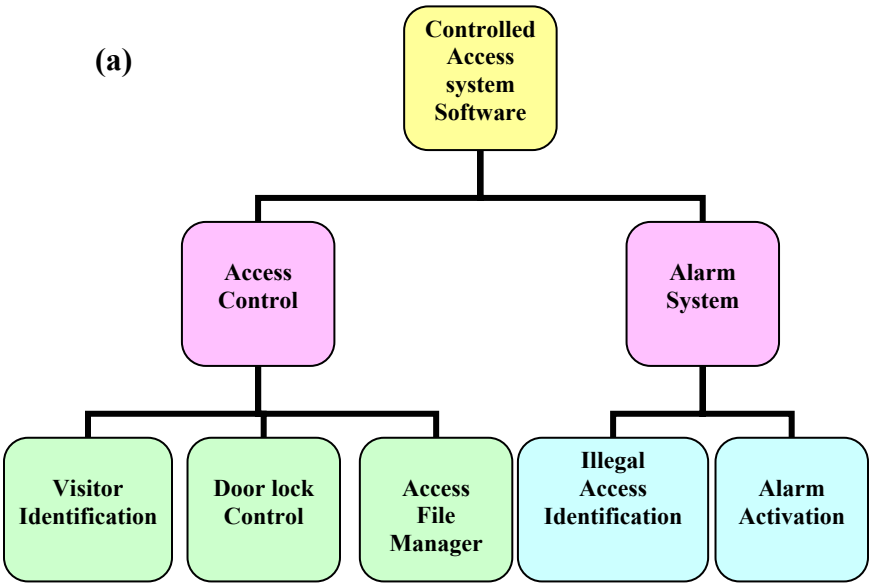
Figure 2(b) describes the same Controlled access system software, but this time it is represented as a hierarchical chart. Here, each component in the diagram represents a real software component.

The System executive main Loop component calls three other components:

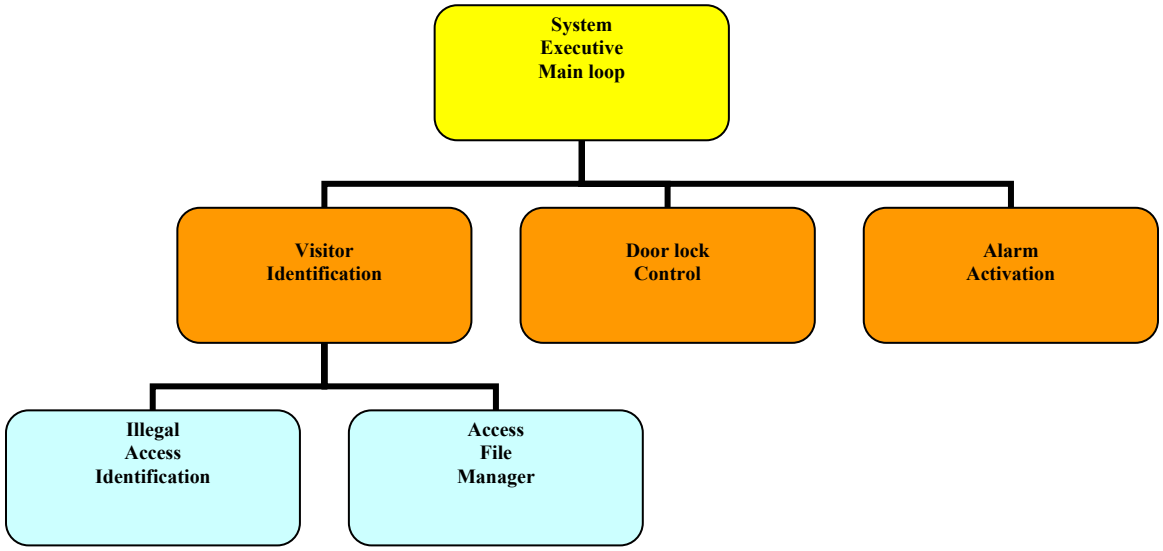
Visitor identification, Door lock control and Alarm activation: The Visitor identification component calls two components: *Illegal access identification and Access file manager.*

Figure 2: ***(a)*** Decomposition of high level components into low level components; (b) a hierarchical structure chart

(a)



(b)



Just like any other large complex task, the development of a software project is more easily managed with the divide and conquer approach.

Stepwise refinement, when applied to a software project, produces all the low level work tasks and includes

- development tasks
- managerial tasks
- support tasks and
- administrative tasks

The WBS list of project tasks is derived from the project's statement of work (the SOW) that defines the scope of the project. The SOW is usually prepared before the official launching of the project, and is often part of the project contract between the customer and the developer.

For internal projects, when an organization is funding its own development work, the SOW becomes synonymous with the Project definition specification or a similar document that defines the scope of work for the software project manager.

b) Functional Decomposition

The functional decomposition of a software project is a division of the system into its operational components as they are seen by the user. Functional decomposition is part of the requirements phase of a project. The objective of this phase is to define air the characteristics of the system from the user's perspective.

Let us consider an automatic bank teller system. The ability to communicate on-line between the remote automatic tellers and the bank's central computer in order to provide updated account information is a functional characteristic or the system.

This will usually be defined during the requirements phase of the development cycle. However, the method of transmission between the automatic teller and the central computer is not a functional characteristic of the system, as this is internal to the design and implementation of the system and is not apparent to the user. The method of transmission, including the communications protocol, will usually be defined during the design phase of the development of the system.

Figure 3 presents an example of the functional decomposition of an automatic bank teller system into lower levels of functional components. In Fig. 3 we have determined that there will be a customer data base, which could be viewed as a design decision.

This is unavoidable. The functional decomposition is rarely completely devoid of all design considerations. As we will see, the functional decomposition is often a starting point for the initial design of the system.

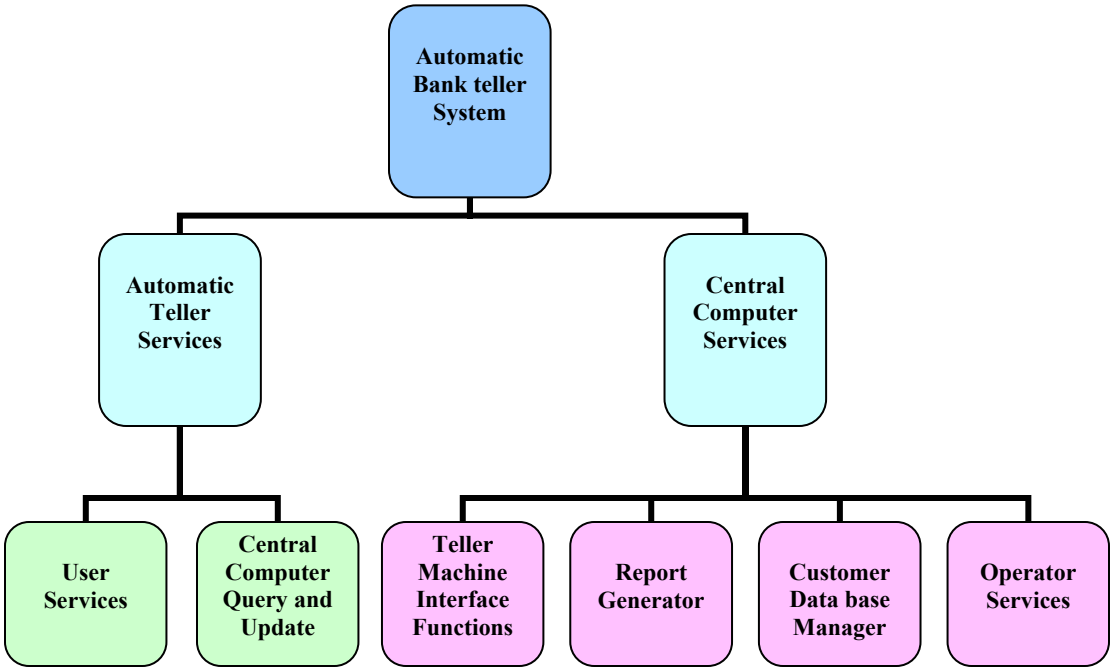


Figure 3: Automatic teller system - functional decomposition diagram

Just as the requirements phase precedes the design phase, so the functional decomposition of a software system will usually precede the design decomposition.

The functional decomposition will often provide much of the information necessary for the subsequent division of the system into the implementation components.

In fact, the functional decomposition is often a good place to start when designing a software system, as the major functional components of a system will often correspond to the initial division of the system into subsystems or high level components.

c) Design Decomposition

The design decomposition of a software system is a division of the system into lower level components that coincide with the actual software components of the system. In a full design decomposition of a software system, the lowest components correspond to programming modules (usually procedures, subroutines or program functions).

The work breakdown structure (WBS) is the decomposition of a software project into low level work tasks.

An important point to remember is that in design decomposition, only the lower level components are actually implemented. Higher level components represent a group of lower level components.

Design decomposition basically produces two types of system component:

- high level components and
- low level modules

Different software development standards use different terminology to identify the various levels of decomposition.

Figure 4 presents an example of the design decomposition of an automatic bank teller system into lower levels of design components. On the third level, the Automatic teller component decomposes into the Hardware interfaces, and the Teller logic. The next level may then decompose the Hardware interfaces into the Keyboard driver, the Display driver, the Printer, driver and the Beeper. At this level, these drivers may represent actual software modules.

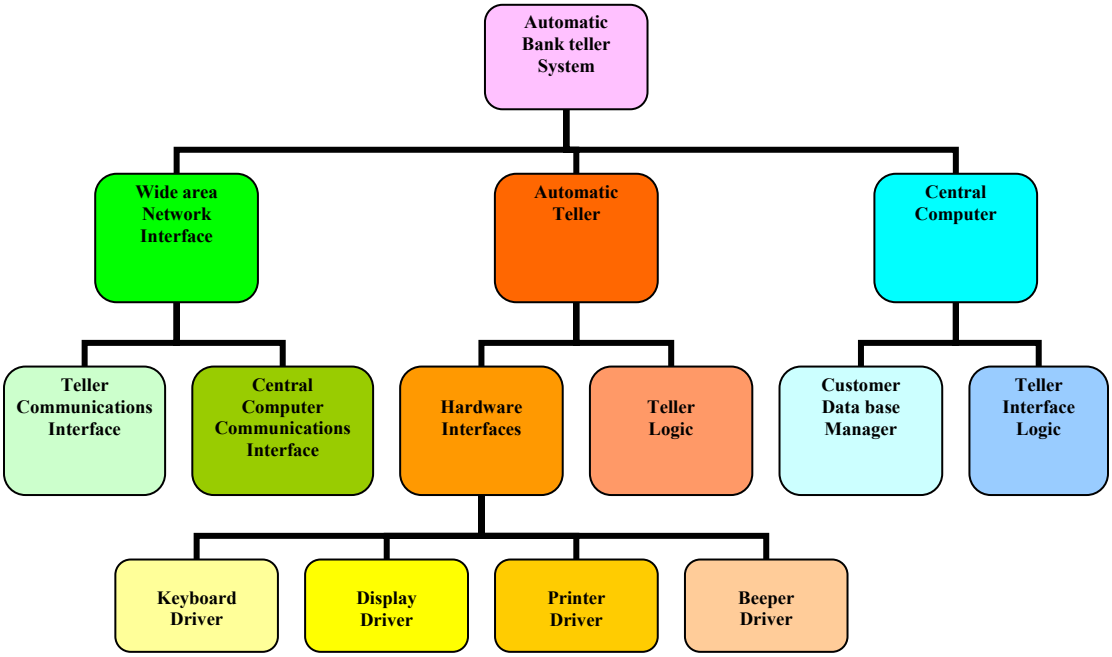
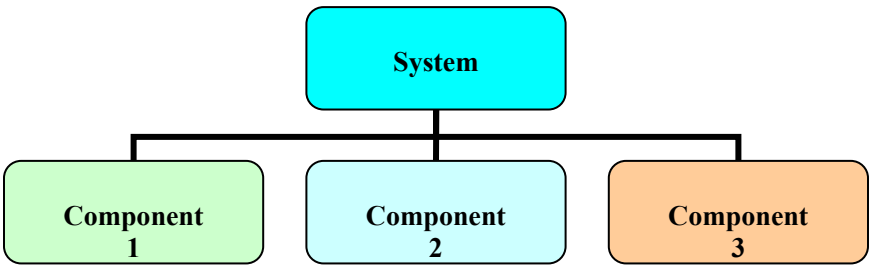


Figure 4: Automatic teller system -design decomposition diagram

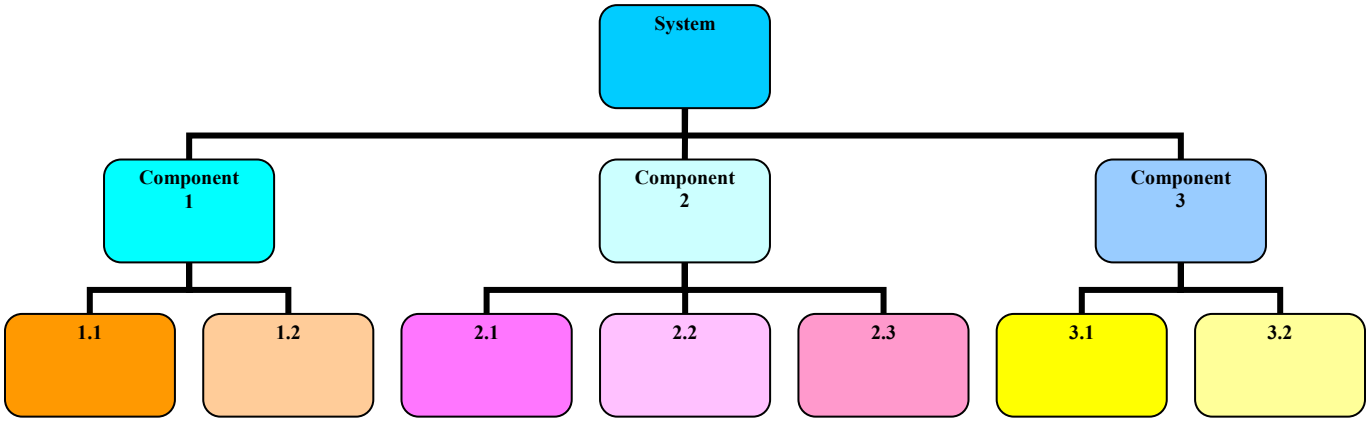
A fully decomposed system, with all its low level components, is not always easy to grasp. This is especially true during the presentation of the system at a project review, when the system needs to be quickly understood by people who have not been involved in its design.

On such occasions, the stepwise refinement technique is a convenient method for gradually presenting progressive detail by initially showing the first decomposition level, and then slowly revealing subsequent levels. This is demonstrated in figure 5.

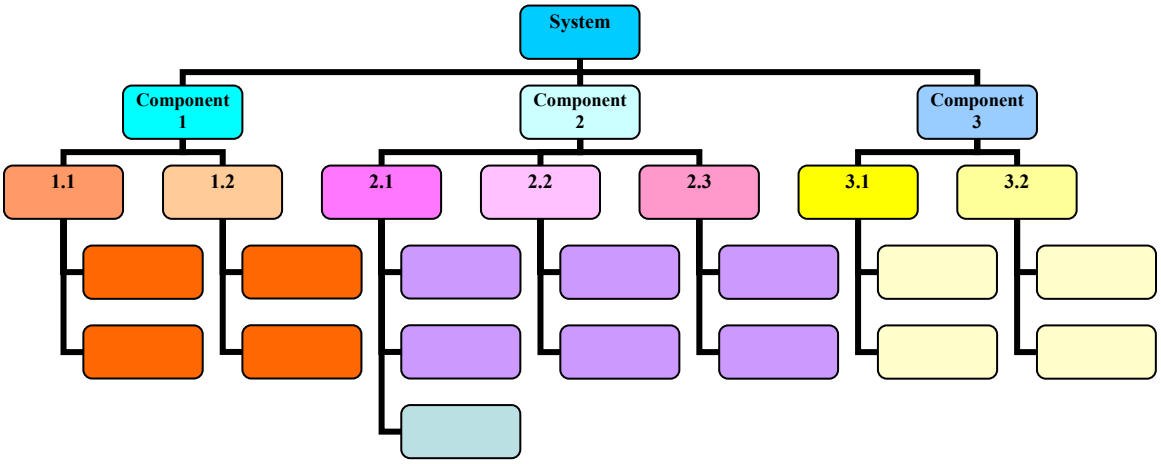
At a convenient intermediate decomposition step we can divide the design in two: the upper levels and the lower levels. This is used particularly when the design phase is implemented in two distinct stages: top level design and detailed design (see Fig.5).



Initial top level decomposition



Intermediate level decomposition – top level design



Low level decomposition – detailed design

Figure 5: Software decomposition by stepwise refinement

d) Develop Project Tasks

WBS tasks are developed by asking, "What tasks need to be done to accomplish the project objective?" The choice of WBS structure is subjective and reflects the preferences and judgment of the project manager.

As levels of the WBS become lower, the scope, complexity, and cost of each subtask become smaller. The lowest level tasks, or work packages, are independent, manageable units that are planned, budgeted, scheduled, and controlled on their own.

As efforts of similar scope and type are planned, the basic WBS tasks remain fairly similar, but each project requires a specific set of tasks that address the uniqueness of the project's requirements. Certain top level elements, such as project management, are included in the WBS of every project, regardless of its type, size, or complexity. Other items, like installation, may not apply to every project.

The initially developed WBS evolves over the course of the planning. It is highly probable that it will look quite different as the scheduling, estimation, and resource allocation portions of the plan are completed.

One of the difficult parts of talking about IT projects generically is the wide range of such projects. Typically, in a small project, there is a single project development phase.

In large or complex systems, however, there are often multiple development phases, where different functional requirements are met.

Sometimes these phases are driven by the need to achieve certain levels of functionality prior to the availability of the complete system.

Other times, the phases are defined to partition the development effort and to reduce the risks associated with larger project efforts.

LECTURE # 36

7. Work Breakdown Structure

7.8 WBS- Major Steps

e) Define Project Development Phases

For large systems, the decomposition of the system into smaller components needs to be done early in the planning cycle.

The rationale for the decomposition must be known, otherwise, different results derived from different reasons for the system decomposition may occur.

For example, if a phase is defined to accommodate user needs, the phase may cross multiple functional areas of the system. If, on the other hand, a system is divided into phases simply to reduce risk, a functional division might occur where the phases represent completion of entire functional areas of the system.

The way in which the phases are handled, differs with each application. Often, phases are handled as top level WBS elements, with tasks associated with each phase defined.

f) Define Task Relationships

If a project is broken down into phases, be sure that the WBS reflects this. The WBS structure denotes a hierarchy of task relationship.

Subtask completion eventually rolls up into task completion, which ultimately results in project completion.

There can, however, also be relationships between tasks that are not within the outlined hierarchy.

These relationships need to be noted, and the ultimate structuring of the tasks optimized to favor a minimum of horizontal dependencies and relationships. If the tasks are not organized efficiently, it becomes difficult to schedule and allocate resources to the tasks.

The project scope of work is an iterative process that is generally done by the project team with the use of a Work Breakdown Structure (WBS), allowing the team to capture and then decompose all of the work of the project.

A WBS is a deliverable-oriented grouping of project components that organizes and defines the total scope of the project; work not in the WBS is outside the scope of the project.

As with the scope statement, the WBS is often used to develop or confirm a common understanding of project scope. Each descending level represents an increasingly detailed description of the project deliverables.

A WBS is normally presented in chart form, however, the WBS should not be confused with the method of presentation—drawing an unstructured activity list in chart form does not make it a WBS.

g) Defining Deliverables

Deliverables associated with each task are shown in the WBS and are reflected in the Deliverables portion of the Project Plan. A sample of a Deliverables template is shown next. All deliverables are listed as they are identified.

As the schedule is completed, the due date is filled in, and responsibility for the deliverable is assigned as it is known (typically when the organization chart is defined). The date delivered is a field that is filled in as deliveries are made.

Over the course of the project, a comparison of the due date and the date delivered provides a metric for how well deliverable dates are met by the project team.

While the deliverables list is a compilation of information identified in the WBS and the project schedule, it is useful to maintain a separate list since deliverable completion can be a key metric of project progress. Separate tracking of deliverables can help keep a project on track. It also serves as a useful communication tool with both stakeholders and the project team.

Product Name	**Due Date**	**Date Delivered**	**Author/ POC**
Requirement Specification	4/1/96	4/1/96	ABC
Design Specification	8/1/96		XYZ
Test Plan	8/1/96		DEF
Implementation Plan	11/1/96		…

Source Code	12/1/96		…
Test Report	1/30/97		…

h) Creating a Work Breakdown Structure

A project can be compared to a large system. A large system consists of multiple, independent subsystems that achieve a common goal. Similarly, a project consists of small, independent tasks.

Each task can be subdivided into sub tasks. Fro example, in a general software project, a task is to perform project analysis. You may also consider studying the organizational objectives and preparing a project proposal to present to the client.

Therefore, in a project analysis task, there are three subtasks. A subtask is also known as a work package. A work package is a unit-level entity in a project system.

You can create a WBS by following the three steps listed below. These are general steps, and they can vary in relation to an individual or an organization.

a) Brainstorm to arrive at board tasks for a project
b) Refine the board tasks
c) Categorize tasks into logical task headers

a) Brainstorming to Arrive at Broad Tasks

The first step in creating a WBS is to organize a meeting of all senior managers, system analysts, and prospective developers. The objective of the meeting is to brainstorm and come up with a set of broad tasks that need to be performed in a project. During the brainstorming session, you can make a note of all possible tasks.

Subsequently, the feasibility of each task is assessed. In addition, any conflict of common tasks can be eliminated. To enable you to perform this activity better, you can arrange tasks as and when they are brainstormed.

For example, XYZ Inc. conducts a brainstorming session to divide the tasks into multiple subtasks that are performed during the analysis phase of a project.

During the session the project managers and the analysts come up with tasks on the basis of prior project experience. These include tasks such as determining the scope of a project, drafting the software specifications, and securing resources for the project.

Some additional tasks are also determined based on client requirements. Preparing the initial project budget and estimating the approximate project timeline are examples of such tasks. In addition, personnel responsible to complete each task are also determined.

The subtasks are subsequently arranged in the order in which they are executed.

Figure 6 depicts the subtasks in the analysis phase. Note that after a questionnaire is prepared, you can either arrange to meet the client or document the feedback collected by the questionnaire. It is clear from the figure that preparation of the SRS document can begin only after the preceding three tasks are complete.

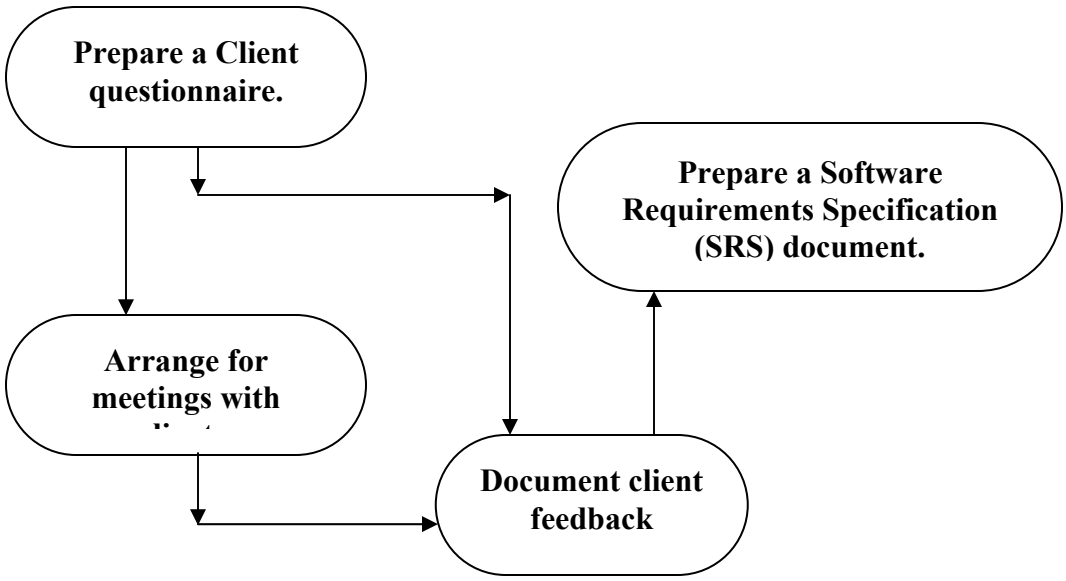


Figure 6: WBS Activity

Dividing a main task into multiple subtasks also enables you to estimate the duration and the effort required for individual tasks.

Subsequently, at the end of the phase, you can evaluate the actual effort and duration. This helps you compare the estimated values with the actual values and prepare a schedule for the subsequent phases.

The duration of each task affects the total duration of a phase. This, in turn, affects the schedule of the subsequent phases. You can make similar deductions while comparing the estimated costs with the actual costs of a task.

b) Refining Broad Tasks

In the second step of creating a WBS, you refine the list of tasks that was compiled during brainstorming.

Refining the tasks may include adding more tasks or combining the existing ones. During this step, you may also change the arrangement of tasks.

A change in the arrangement of tasks can occur on the basis of two theories of WBS.

- *Deliverable-based theory*
- *Project life-cycle-based theory*

You use the deliverable-based theory when the deliverables of a project are more important than its phases. This normally happens when the deliverables are decided before the project begins.

However, if the phases of a project are completely defined, you use the project life-cycle-based theory. You can use the project life-cycle-based theory to arrange the tasks in a project.

c) Categorizing Tasks into Logical Headers

Finally, after defining tasks and arranging them, you categorize each task into a logical task header.

For example, preparing test plans and test cases and drawing up the test plans can be categorized as *Testing.*

This activity provides another chance to ensure that you have not missed any task during brainstorming.

In addition, you can also consult an expert such as a senior manager to review and validate the tasks identified.

7.8 WBS Implementation

i. Implementation Strategy

When you set up a project WBS, think about how you will be using it later in the project. Consider how you will organize the WBS, schedule format, manager assignments, and charge numbers, in your early project planning. It will be helpful if you can map the charge numbers, managers, and task groups to each other. This will help you track costs and progress for each manager.

If your project schedule will on MS-Project, you may want to insert "text" columns into your schedule (Gantt View) for project charge numbers and manager names.

Some project management environments have definite conventions for grouping items in a WBS. The best method is to have a WBS that works for your particular project environment. The WBS should be designed with consideration for its eventual uses.

Your WBS design should try to achieve certain goals:

- *Be compatible with how the work will be done and how costs and schedules will be managed,*
- *Give visibility to important or risky work efforts,*
- *Allow mapping of requirements, plans, testing, and deliverables,*
- *Foster clear ownership by managers and task leaders,*
- *Provide data for performance measurement and historical databases, and*
- *Make sense to the workers and accountants*

There are usually many ways to design a WBS for a particular project, and there are sometimes as many views as people in the process.

Experience teaches that everyone takes a slightly different slice of the apple, so make sure WBS arguments seeking metaphysical certainty are quickly brought to closure.

ii. Methodology

- PM must map activities to chosen lifecycle as each lifecycle has different sets of activities.
- Integral process activities occur for all Planning, configuration and testing.
- Operations and maintenance phases are not normally in plan (considered post-project)
- Some models are “straightened” for WBS:
  - *Spiral and other iterative models*
  - *Linear sequence several times*
- Deliverables of tasks vary by methodology

7.9 Guidelines for decomposition of work tasks

i. General Guidelines

- A WBS should be easy to understand. Some companies have corporate standards for these schemes. Some top-level items, like Project Mgmt. are in WBS for each project. Others vary by project

- What often hurts most is what's missing. Break down until you can generate accurate time & cost estimates. Ensure each element corresponds to a deliverable.

- *How detailed should it be?*
  - Not as detailed as the final MS-Project plan
  - Each level should have no more than 7 items
  - It can evolve over time

- *What tool should you use?*
  - Excel, Word, Project
  - Org chart diagramming tool (Visio, etc)
  - Specialized commercial apps

- Re-use a "template" if you have one. Although each project is unique, WBS can often be "reused" since most projects will resemble another project to some extent.

- **Ex:** Most projects within a given organization will have the same or similar project life cycles, and will thus have the same or similar deliverables required from each phase.

- Many application areas or performing organizations have standard or semi-standard WBS s that can be used as templates.

- **Ex:** Financial Management System etc
- Identify major elements of the project. In general, the major elements will be project deliverables and project management.

- Decide if adequate cost and duration estimates can be developed at this level of detail for each deliverable.

- Identify constituent components of the deliverable. Constituent components should be described in terms of tangible, verifiable results to facilitate performance measurement.

- Verify the correctness of the decomposition: are the low-level items both necessary and sufficient for completion of the decomposed item? Is each item clearly and completely defined? Can each item be appropriately scheduled and

budgeted?

- The WBS is not a decomposition of the software produced by the project. It is a decomposition of the project itself, and includes such activities as management, procurement, installation and, of course, software development.

- Each low level module may be assigned three basic work tasks:
  - *module design,*
  - *coding and*
  - *unit testing*

- Additional development tasks such as prototyping, testing, and integration are derived from the other development phases.

ii. A typical list of high level WBS tasks

Following table contains a typical list of high level WBS tasks to be included in the formal WBS task list.

- This is not an exhaustive list of all project development tasks, and not all projects will require all the tasks described.

- However, this table will be useful as a checklist to assist in locating tasks that may have been overlooked.

Table: High level WBS structure tasks

Software development
- **Requirements analysis**
- **Prototype development**
- **Prototype specification**
- **Prototype design**
- **Prototype implementation**

Design
- **Top level design**
- **Detailed design**

Implementation
- **Coding**
- **Unit test**

Integration
- **Software integration**
- **Hardware/software integration**

Testing
- **Alpha testing**
- **Beta testing**

Acceptance

Installation / Maintenance

Error correction

Software enhancement

Management

Planning

Staffing

Administration and services

Budget administration

Personnel management

Quality assurance

Configuration management

Training / Procurement

Acquisition of development tools

Acquisition of system components (off the shell)

Equipment selection

Vendor selection

Ordering procedure

Inventory control

Documentation

Technical writing

Project publishing activities

Development documentation

Non-deliverable development documentation

Deliverable development documentation

Maintenance documentation / User documentation

iii. Management and Administration Tasks

- Non-development activities, such as high level management WBS tasks, are standard, to a large extent, and any variance is determined by either the magnitude of the project, or the introduction of a new management model.
- An example of a list of management WBS tasks appears in the following Table.
- This list contains many of the most important management tasks required for most software development projects.
- Those tasks that are mandatory for all projects are marked as such in the list.

Table: Management and administration tasks

Mandatory	Management task
√	1. Planning
√	2. Preparation or estimates

√	3. Risk analysis and risk management
√	4. Scheduling
	5. Staffing
	6. Budget analysis and administration
√	7. Personnel management
√	8. Task assignment
√	9. Delegation of authority
√	10. Assignment of development resources
	11. Supervision of development equipment maintenance
	12. Supervision and control of development
	13. Organization of reviews and formal presentations
	14. Establishment of standards and methods
√	15. Quality assurance and control
√	16. Configuration management and control
	17. Supervision of subcontractors and vendors
√	18. Higher management interface and coordination
	19. Customer interface and coordination
√	20. Reporting
	21. Administration and services

- Note that budget analysis and administration is not a mandatory management task, simply because not all projects administer their own budget. Some organizations have a financial officer responsible for the administration of project budgets.
- Customer interface is also not a mandatory management task because not all projects have a customer. In the case of company internal projects, high level management, together with the designated users of the system being developed, play the role of customer. It is usually they who specify the initial project requirements, and it is to them that the project manager must come for final approval and for final system acceptance.

- Break down the work until accurate estimates of cost and resources needed to perform the task are provided.

- Ensure that clearly defined starting and ending events are defined for the task. This may be the production of a deliverable or the occurrence of an event.

- Verify that the lowest level tasks can be performed within a "reasonable" period of time. Each state organization must define "reasonable."

- If the time period to complete a task is too long, an accurate project status in the implementation phase may not be possible.

- An industry standard rule of thumb is to make work packages that can be

completed in timeframes of two weeks.

- Verify that people assigned to the project are all assigned a WBS task. Have a firm rule: if the task is not on WBS, it is not worked on.

- The work breakdown structure and any support documentation should be easy to understand.

- The work should not be subdivided arbitrarily to the lowest possible level. Work breakdown structure elements at the lowest control level should typically range from 0.5% to 2.5% of total project budget.

- Furthermore, you should be aware that the work breakdown structure can be developed to reflect the trust that you have in specific line groups, by leaving them the autonomy over specific areas of work.

- Finally, always remember that projects are dynamic working environments - so try to maintain flexibility in the work breakdown structure.

- As a general guideline, no single functional decomposition should be selected just because it was conceived first. The functional decomposition of a software system may be substantially different from the design decomposition of the system. However, a good functional decomposition will have taken into account design as the next development phase, and will often be a good starting point for the division of the system into high level design components. The high level design components are then further decomposed into successive lower levels that ultimately produce programming modules. A good modular design produces small, simple, independent modules.

LECTURE # 37

8. Scheduling

8.1 Understanding Schedule

i. Salient Features:

Fred Brooks, the well-known author of *The Mythical-Man-Month* [BRO95], was once asked how software projects fall behind schedule. His response was as simple as it was profound: "One day at a time."

The reality of a technical project (whether it involves building a hydroelectric plant or developing an operating system) is that hundreds of small tasks must occur to accomplish a larger goal.

Some of these tasks lie outside the mainstream and may be completed without worry about impact on project completion date, Other tasks lie on the "critical" path, these "critical" tasks fall behind schedule, the completion date of the entire project is 'put into jeopardy,

The project manager's objective is to define all project tasks build a network that depicts their interdependencies, identify the tasks that are critical within the network, and then track their progress to ensure that delay is recognized "one day at a time." To accomplish this, the manager must have a schedule that has been defined at a degree of resolution that enables the manager to monitor Progress and control the project.

Software project scheduling is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks.

It is important to note, however, that the schedule evolves over time. During early stages of project planning, a *macroscopic schedule* is developed. This type of schedule identifies all major software engineering activities and the product functions to which they are applied.

As the project gets under way, each entry on the macroscopic schedule is refined into a *detailed schedule.* Here specific software tasks (required to accomplish an activity) are identified and scheduled.

Scheduling for software engineering projects can be viewed from two rather different perspectives. In the first, an end-date for release of a computer-based system has already (and irrevocably) been established.

The software organization is constrained to distribute effort within the prescribed time frame. The second view of software scheduling assumes that rough chronological bounds have been discussed but the end-date is set by the software engineering organization. Effort is distributed to make best use of resources and an end-date is defined after careful analysis of the ware. Unfortunately, the first situation is encountered far more frequently than the second.

As a project manager, you need to assign durations to all activities and monitor their progress.

You also need to plan the order in which the activities will be performed as well as the start and end dates for each activity. In short, you need to create a project schedule.

The plan includes not only the scheduling of development activities, but also the scheduling of project resources, particularly people. To accomplish this, the manager must have a schedule that has been defined at a degree of resolution that enables the manager to monitor Progress and control the project. The project schedule is one of the most important parts of the project development plan.

The project schedule reflects the start dates and end dates for each activity in the project. It also reflects the resources required to complete the activities. To create a schedule, you can use project management tools.

To create a schedule, you can use project management tools, e.g.:

- Gantt charts
- Network-scheduling techniques

No matter how well the project schedule is prepared, that schedule is useless unless it is adhered to. It is the project manager's responsibility to withstand pressure and to assure that the project is developed in an orderly fashion, according to the schedule. Whenever circumstances change, the project schedule should then be updated to reflect the new situation.

ii. Need for Project Scheduling

Software project have a tendency to get out of control because of the multiple activities that need to be monitored, tracked, and controlled. When a project goes out of control, the original deadlines, the budget, and the effort required overshoot the initial estimates. This not only impacts the product, but also the credibility of the development team.

iii. What Delays Software Projects

There are various reasons why a software project is delayed. Some of the reasons are listed below:

- The initial estimate of the effort and resources required to complete the software project was incorrect. (An honest underestimate of the amount of effort and/or the number of resources that will be required to do *the* job).
- Changing customer requirements that are not reflected in schedule changes.
- Known risks were not mitigated. (Predictable and/or unpredictable risks were not considered when the project commenced).
- Technical difficulties surfaced during software product development. (Technical difficulties could not have been foreseen in advance).
- Human difficulties, such as interpersonal problems and team attitude, became unmanageable. (Human difficulties that could not have been foreseen in advance, e.g., Miscommunication among project staff resulting in delays).
- A failure by project management to recognize that the project is falling behind schedule and a lack of action to correct the problem.
- The initial deadline set for the project was unrealistic. (An unrealistic deadline established by someone outside the software development group and forced on managers and practitioners within the group).

Aggressive (read “unrealistic") deadlines are a fact of life in the software business Sometimes such deadlines are demanded for reasons that are legitimate, from the point of view of the person who sets the deadline. But common sense says that legitimacy must also be perceived by the people doing the work.

iv. How to Prevent Delays

While most of the above reasons result in project delays, unrealistic deadlines cause the maximum damage. Unrealistic deadlines are caused either by the client wanting a quick delivery or the management wanting a return business nom a client.

However, unrealistic deadlines are the norm for software projects. You need to resolve the conflict between an unrealistic deadline and the constraints of the software development team.

As a project manager you need to balance the limited resources and the final deliverable committed to the client. To do this, you can follow the points listed below:

a) Study similar past projects for the method of dealing with similar situations. Use the historical information to project time and effort estimates for the current software project.
b) Use an incremental process model to create a schedule to deliver the required functional module of the software product.
c) Present the schedule created on the basis of past projects to the client. Explain the projected delays and the reasons for the same.
d) Present the incremental model as a strategy for developing the software product.

All the strategies have their advantages and disadvantages. If the schedule created using inputs from the past project does not help meet the deadline, you can use the incremental model. In the incremental model, the strategy is to deliver each module as and when it is completed. However, historical trends show that clients opt for either strategy if provided reasonable time and effort estimates.

8.2 Scheduling Fundamentals

i. Scheduling Basics

Only a small percentage of all the activities in a software project have a direct impact on the on-time completion of the software project. Therefore, you need to identify the activities that are critical to the completion of the project. In addition, you ensure that all inputs are available to complete the critical activities. This is where a software project schedule comes into the picture.

The goal of a software project schedule is to determine the duration of the software project and the phases within the project. A software project schedule enables you to distribute the estimated effort to be spent in performing the critical activities.

There are a few basic principles that determine how a software project schedule is created. To create a project schedule, first you group similar activities together. Then, you determine the dependencies of the different activities. Next, you allocate the estimated time and resources to each activity. Finally, you define the roles and responsibilities and the output and validation criteria.

Some of the guidelines for creating a software project schedule are discussed below:

a) **Classification**
b) **Interdependence**
c) **Time and effort allocation**
d) **Validation criteria**
e) **Defined responsibilities and outputs**

a) Classification

While managing a software project, you need to group similar tasks and activities so that they are completed successfully. The primary tools used to group similar tasks are the WBS and decomposition technique. Using these tools, you can divide a software project into different phases. The phases can be further subdivided into activities. The software project schedule is prepared according to the arrangement of the phases.

b) Interdependence

As you have seen, a software project is composed of multiple phases and each phase is composed of multiple activities. Although each activity is treated separately, it is linked to other activities. As a project manager, you determine the interdependence and sequence of activities. For example, some activities can be completed without any inputs from other activities, whereas other activities cannot start unless a preceding activity is completed.

c) Time and Effort Allocation

Each activity in a software project needs a certain amount of time and effort for completion. To manage the project, you assign start and end dates to each activity. You also need to allocate appropriate effort to each activity. Most software projects operate with time and effort constraints. Therefore, managing within the available resources is very important for a software project manager.

d) Validation Criteria

You also determine the validation criteria for time and effort allocation in a software project. Determining the validation criteria allows you to ensure that the optimal level of resources is available for a particular activity. Suppose 5 people are assigned to an activity that requires an effort of 3.5 person days. This means resources allocated are more than the actual requirement.

e) Defined Responsibilities and outputs

As a software project manager, you assign roles and responsibilities to all people assigned to a software project. This defines the hierarchy in the development team. You also define the outputs from each activity. This helps in identifying the results expected at the end of every activity. Next, you link the roles to the outputs. When each role is linked to the expected results, you

can track each person's effort and the progress of each activity towards closure.

LECTURE # 38

8. Scheduling

8.3 Scheduling Tools

i. GANTT CHARTS

There are various tools that help you create a schedule. One of the simplest project management tools used to represent the timeline of activities is the Gantt chart.

Long before the advent of computers, Henry L. Gantt lent his name to a simple and very useful graphical representation of a project development schedule.

The Gantt chart shows almost all of the information contained in the schedule activity list, but in a much more digestible way. The schedule information is more easily grasped and understood, and the activities can be easily compared. The Gantt chart enables us to see at any given time, which activities should be occurring in the project.

A Gantt chart has horizontal bars plotted on a chart to represent a schedule. In a Gantt chart, you plot time on the horizontal axis and activities on the vertical axis.

You represent an activity by a horizontal bar on the Gantt chart. The position of a horizontal bar shows the start and end time of an activity and the length of the bar show its duration. You can have one look at the Gantt chart and make out the progress of the project. Figure 1 displays a sample Gantt chart.

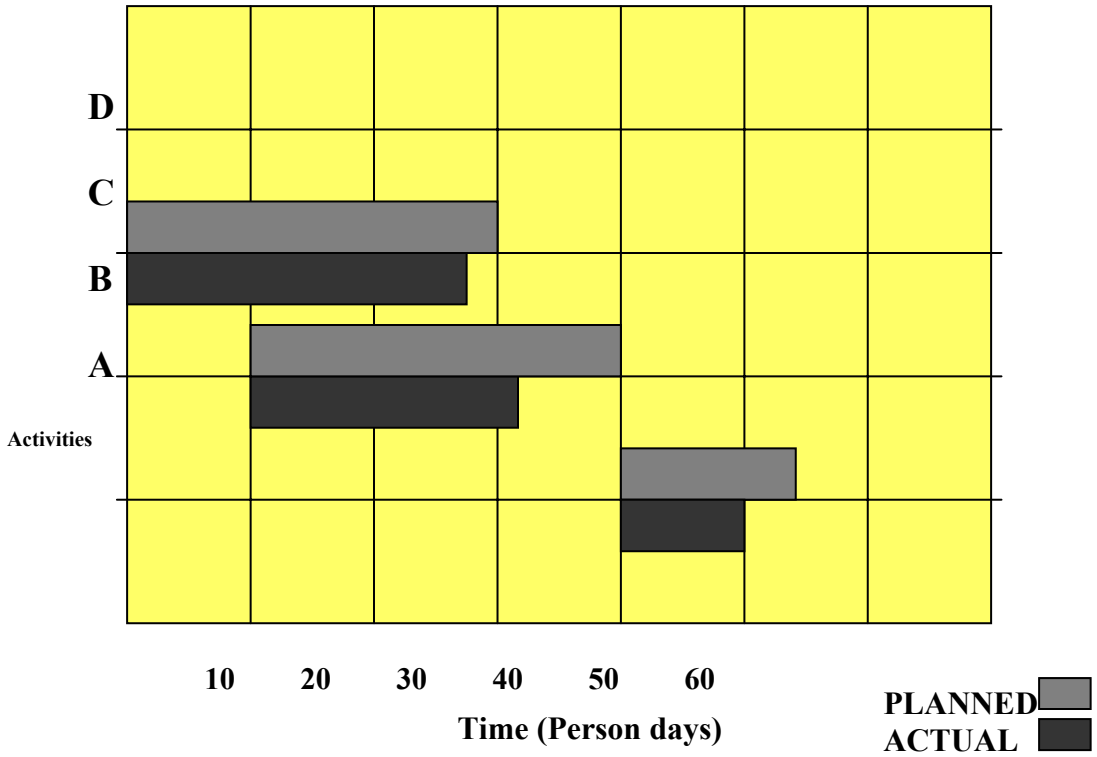


Figure 1: Example Gantt Chart

In Figure 1, activity D starts on day 1 of the project. The planned duration of this activity is 20 days. The planned duration is shown in a lighter shade on the Gantt chart. This is done to differentiate planned duration from the current status of the activity. In the case of activity D, the current duration of the activity is 19 days. Therefore, activity D is still one day short of completion. This can be discerned from the length of the gray and black bars. Activity C, planned for completion on day 40, is much behind schedule. This can be observed from the smaller length of the black bar in the Gantt chart.

To understand how you can use a Gantt chart to schedule a project, consider an example. Table 1 display a set of activities in a software project and the start and end time for each activity.

Table 1: Project Activities and Time Allocation Details

Tasks	Time Allocation (person days)	Planned Start	Planned End
Requirement analysis and project planning	10	01/01/2002	01/14/2002
Setting up the environment	6	01/01/2002	01/08/2002
Software construction	80	01/15/2002	05/15/2002
Unit testing	30	05/16/2002	06/28/2002
User training	5	07/22/2002	07/26/2002
System testing	15	07/01/2002	07/19/2002
User documentation	30	01/15/2002	02/28/2002
Data migration	20	01/15/2002	02/15/2002
Conducting user acceptance test	20	08/01/2002	08/31/2002

Using the data shown in the Table 1, you can create a Gantt chart. The Gantt chart is displayed in Figure 2.

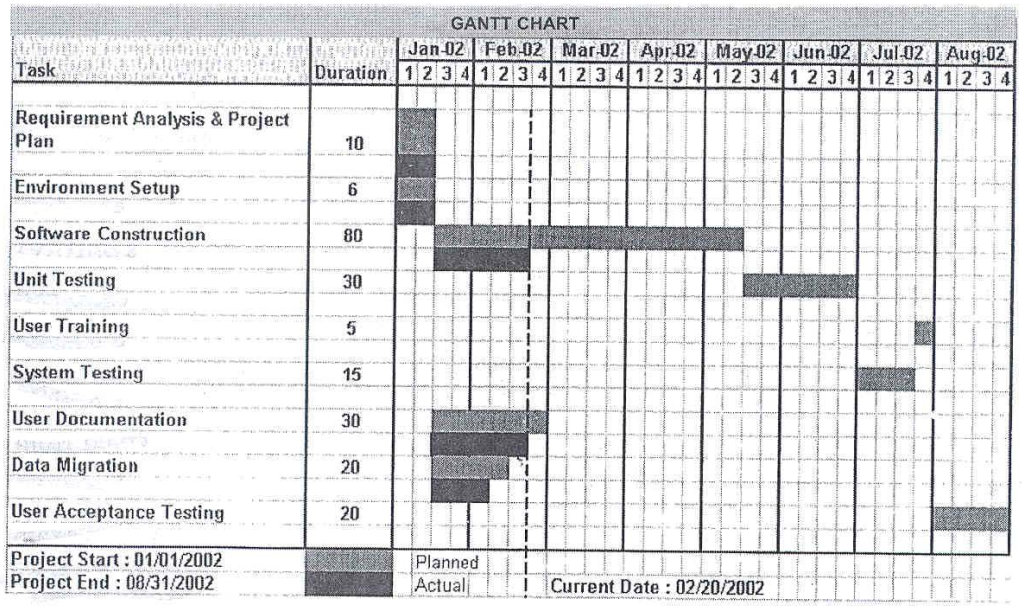


Figure 2: Gantt Chart for Activities of Table 1

In Figure 2, the Gantt chart displays the week-wise status of a software project. A vertical dotted line is drawn through the chart to represent the current date. This

line indicates the status of the project on a specific day. The left side of the line indicates the tasks that are completed. On-going activities run across the line. The future activities lie completely to the right of the line. After drawing the Gantt chart you can extend the bars every week to a length proportional to the work completed during the week. Therefore, the bars describe the status of the-project at any point of time.

This process of creating a Gantt chart as discussed above is manual. Nowadays, automated tools are available for creating Gantt charts. For example, you can use Microsoft® Project™ 2000 developed by Microsoft to create a project schedule automatically.

Figure 3 is another example of a Gantt chart. The symbols used in the chart are widely accepted, though not standardized. The inverted triangle, for example, is commonly used to represent a significant event, such as a major milestone.

The Gantt chart in Fig. 3 demonstrates the ease with which important schedule information can be quickly perceived. We can immediately see that, except for the maintenance phase, all phases overlap, and that from November to mid-December 1992 three high level activities overlap.

Major milestones

SRR = Software requirements review
POR = Preliminary design review
COR = Critical design review
TRR = Test readiness review
ATP = Acceptance test procedure

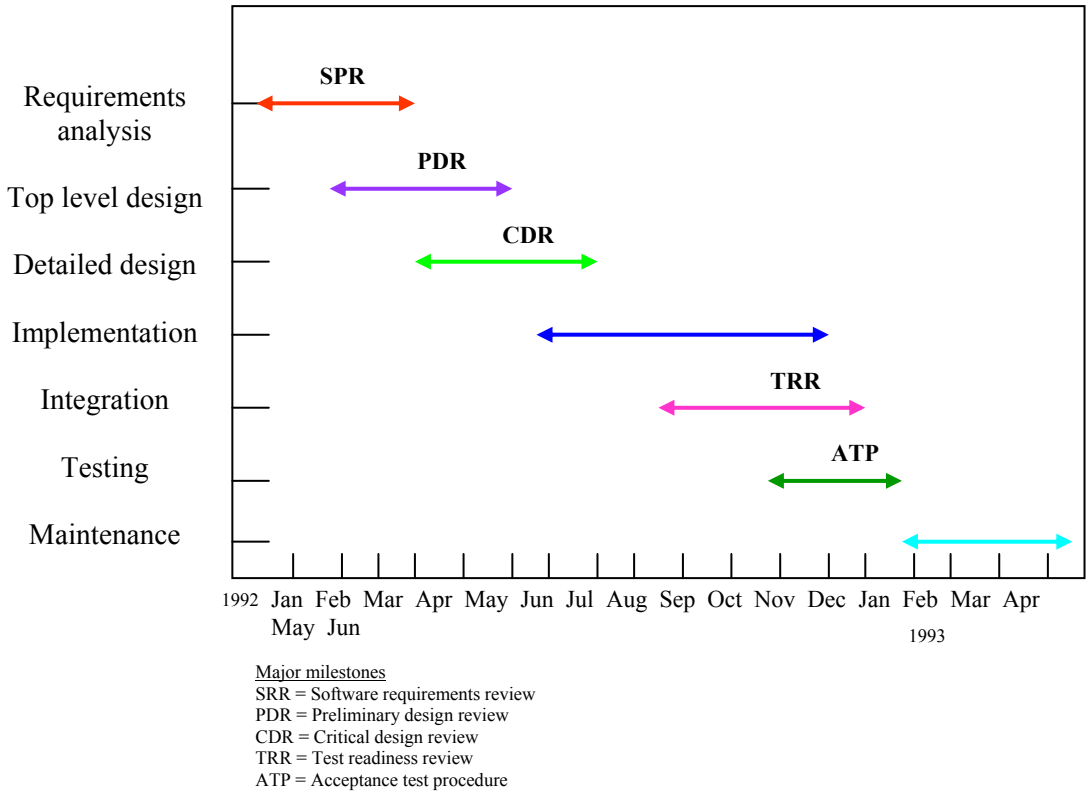


Figure 3: High level Gantt chart -project development schedule.

More detailed charts can also include the names of the engineers assigned to each activity, and the equipment that will be needed for each activity. This information can be added next to the activity time lines in the graph, or as an inserted reference table (similar to the list of major milestones in fig 10.1). Some variations of the Gantt chart do include this type of information on the chart, but this can cause clutter, which is contrary to the main objective of the chart; to enable important schedule information to be grasped quickly.

It is also important to understand what Gantt charts do not provide. In a Gantt chart, it is difficult to provide information on the amount of resources required to complete each activity. A common mistake is to conclude that if five engineers are assigned to integration, and the integration activity starts in mid-September 1992 and ends in mid-January 1993 (four months), then integration requires 20 work months. In fact, integration may start with only one engineer, with one more joining during the second month, and the remaining three engineers joining during the third month. The integration team may then be reduced to three engineers during the fourth integration month.

Figure 3 includes only seven activities. As more detail becomes available, lower level activities can be included on the chart. When the chart has more activities than it can reasonably carry (a subjective decision), additional charts may be

added. For example, the design activity can be presented on a separate Gantt chart (see Fig. 4).

Figure 4 presents both high and low level activities. For example, 'Integrate phase 1 model' contains three low level activities: 'Integrate executive', 'Integrate operating system' and 'Integrate user interface'. This provides the continuity link between the detailed Gantt chart (Fig. 4) and the higher level chart (Fig. 3).

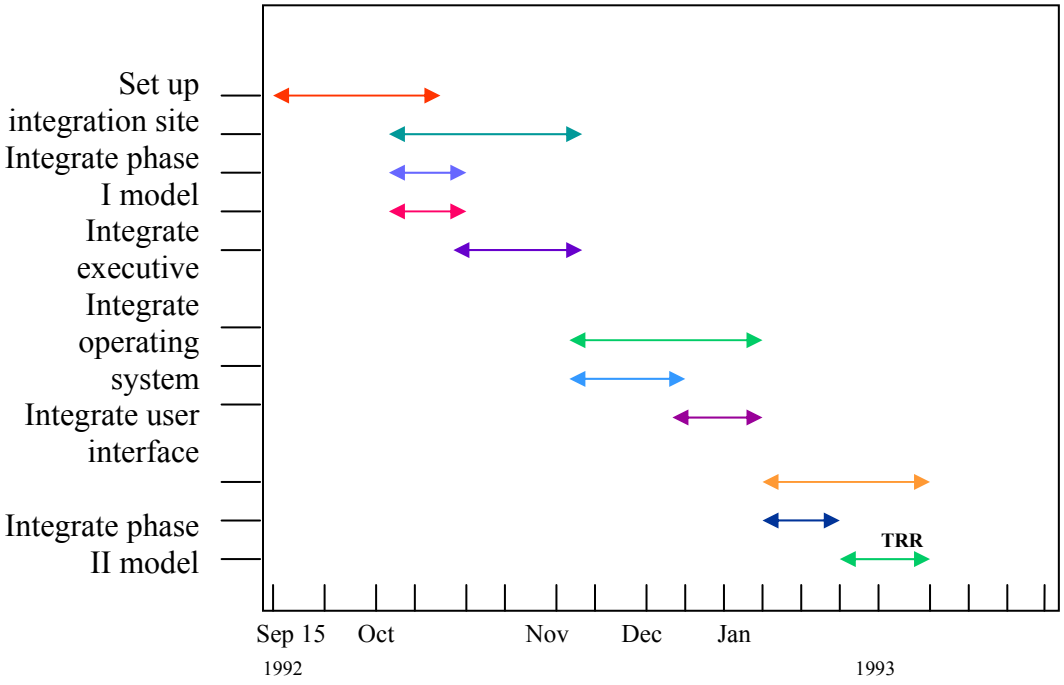


Figure 4: Detailed Gantt chart -integration schedule

Note that each period of one month in Fig. 3 has been divided into four weeks. Though not completely accurate, this is a common approximation, used also in estimation and apart from being convenient; it also provides some slack for minor scheduling adjustments.

Similar detailed Gantt charts can be prepared for each of the major project development phases. Non-development activities will also appear on the Gantt chart, such as 'Procurement of development tools', or 'Market research'. This is particularly useful when certain development activities are dependent on other non-development activities, such as the procurement of development tools (e.g. a compiler) that need to be completed before the implementation activities can begin. In cases where such dependent relationships may have been overlooked, they will often emerge from' a review of the Gantt chart. This type of dependence

between activities is best presented in another type of chart, called a *Network precedence chart* or a *PERT chart.*

Network Scheduling Techniques

To plan the activities in a project, you can also use network-scheduling techniques. Network scheduling techniques use network schedules to trace the completion of predetermined activities.

There are two basic network-scheduling techniques:

- PERT and
- CPM

You can use either of these techniques to analyze a wide variety of projects. Each technique depicts a project as a sequence of activities. This helps you perform an analysis of individual activities or the complete project.

The network-scheduling techniques also enable you to analyze the dependencies that exist between the activities.

Using PERT and CPM, individually or in combination, helps you complete a project on time. By using these techniques, you can determine the latest time by when an activity should start to be completed on time.

Despite the different approaches followed by PERT and CPM, both techniques have some common components. These include:

a) Activities
b) Nodes
c) Network
d) Critical path

a) Activities

Activities are the basic building blocks of network schedules. An activity is defined as a task that consumes time, effort, money, or any other resource. It is necessary to specify all the activities of a project by breaking down a project into several steps.

You need to define the steps in such a way that they are distinct, homogeneous tasks for which you can estimate resource requirements.

Each activity is represented on a network schedule by using an arrow with its head indicating the direction in which the project will progress. Each activity is identified by a description or an alphabet. In addition, the estimated duration of

each activity is placed below the activity. Figure 5 shows an activity with expected duration of 15 days.

15 Days

Figure 5*:* An Activity

After identifying all the activities in a project, it is necessary to schedule them. This enables you to arrange the activities in the order of completion. You sequence the activities based on their types.

The different activities that are possible in a project are predecessor, successor, and concurrent activities.

Predecessor activities need to end before the next can begin. After a predecessor activity is completed, the successor activity becomes the predecessor for another activity. However, unlike the predecessor arid successor activities, the concurrent activities can be completed simultaneously with other activities.

Some activities in a project, irrespective of whether they are predecessor, successor, or concurrent, may depict a float period. Float is the amount of time by which an activity may be delayed without affecting the completion date of the entire project. However, the complex dependencies that exist between activities result in sequencing constraints for projects. To a large extent, these constraints limit the flexibility that you may otherwise have in project planning.

b) Nodes

A node on a network schedule is that point in time at which an activity either begins or ends. The point where an activity begins is called a tail node and the point where it end is called its head node. On a network schedule, a circle represents a node. A number identifies each node in a network schedule. Figure 6 represents the tail and head nodes for the activity of interviewing clients in a project.

Figure 6: An Activity Connecting Two Nodes

c) Network

A network is the graphic representation of the activities in a project. It depicts all the activities and nodes in the project. On- a network, the arrows terminating at a node need to be completed before the following activity can begin. Figure 7 shows a sample network consisting of five activities in a project.

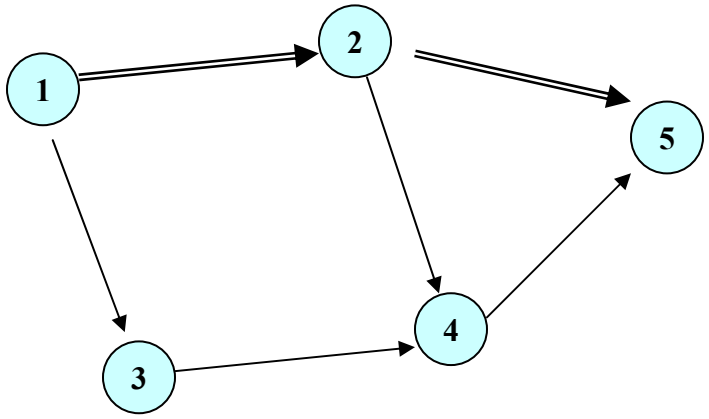


Figure 7: Sample Network Schedule

d) Critical Path

The critical path is the longest path through a network. It consists of those activities that cannot be completed concurrent1y. In other words, the critical path represents the maximum duration for a project. You can determine the maximum duration by adding the duration of each activity on the critical path. Typically, a double line in a network schedule represents the critical path for that project. Figure 8.6 shows the critical path for a network.

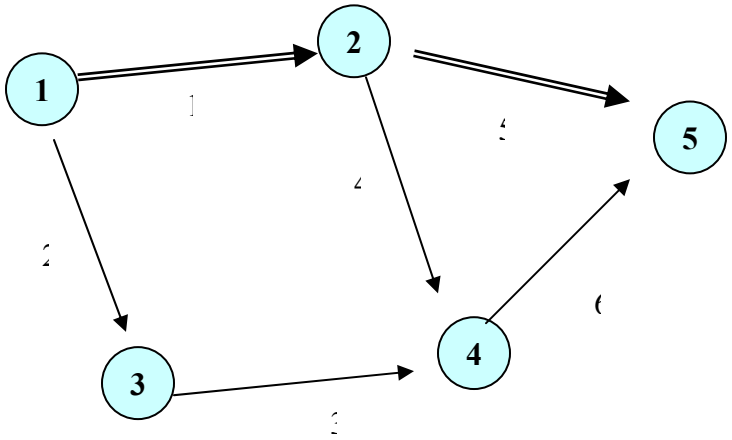


Figure 8: Network Schedule with Critical Path Identified

All the activities on the critical path are critical for a project. If an activity on the critical path is delayed, the entire project is delayed by the same amount of time. You need to monitor the activities on the critical path because the project depends on the successful completion of these activities. If required, additional resources can be applied to these activities to shorten the project duration. However, some of these activities may also depict a float period.

8.6 Rules for Creating a Network Schedule

There are a few basic rules that are followed while constructing a network schedule.

1. Each activity has a preceding node and a succeeding node.

2. Each node has a distinct number. As a convention, the number that is assigned to the head of the arrow is greater than the number that is assigned to the tail.

3. The network schedule has no loops. For example, in Figure 9, activity 1 is the predecessor of both activity 2 and activity 3. This places activities l, 2, and 3 in a loop.

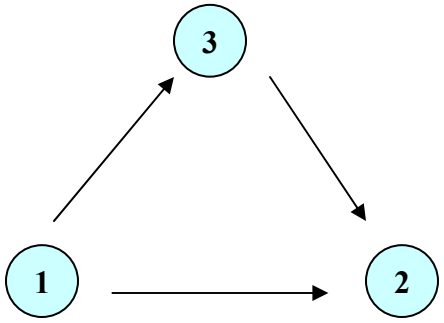


Fig 9: A loop is not permitted

4. Each activity has a unique preceding and succeeding event associated with it. For example, in Figure 10, activities A and B have common preceding and succeeding events associated with them. This is not allowed in a network schedule.

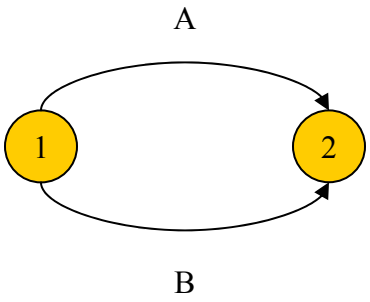


Figure 10: Activities with Common Preceding and Succeeding Events

At times, you need to introduce a dummy activity in a network schedule. This is an imaginary activity that enables the network schedule to display parallel activities. For example, in Figure 11, consider A and B as two parallel activities that can be executed simultaneously. Both A and B must end so that activity C can begin. A dummy activity is introduced as a dashed arrow to mark the start of activity C. The dummy activity is introduced to show the dependency between activities in the network schedule and does not have a description or duration.

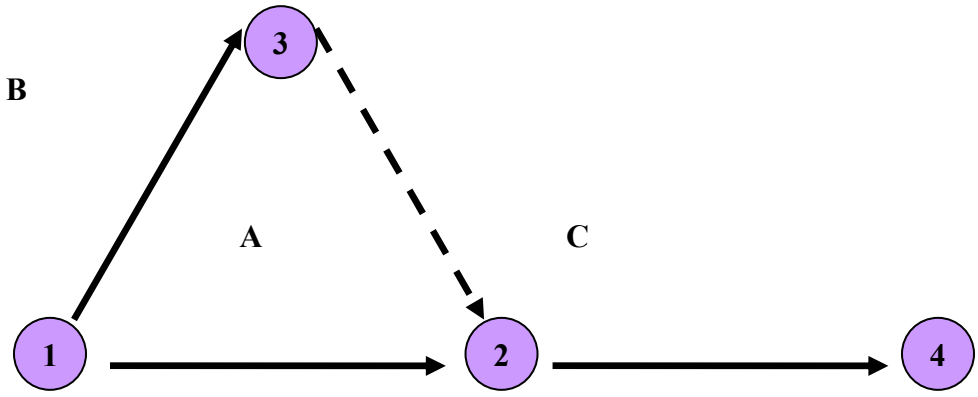


Figure 11: Dummy Activity

8.7 Using PERT to Schedule a Project

PERT was developed in 1957 to cater to the needs of the Polaris Fleet Ballistic Missile project of the US government. PERT uses a probabilistic approach to time estimates. You normally apply it to projects that are characterized by uncertainty.

For example, in a complex software project where you require research to identify activities. PERT allows you to account for the uncertainties that are common to most software projects. PERT uses the network schedule to represent a project schedule while taking the uncertainties into account.

i. Time Estimates in PERT

PERT is a probabilistic technique that uses three time estimates: It assumes that activity times are represented by a probability distribution. To finish an activity, it bases the probability distribution of activity time on three time estimates:

- *Optimistic time*
- *Pessimistic time*
- *Most likely time*

The optimistic time is the shortest time period within which an activity can end if everything goes well.

The pessimistic time is the time that an activity takes to complete if everything that can go wrong goes wrong. This is the longest time that an activity can take to complete.

The most likely time is the estimate of the normal time that an activity takes to complete.

From these three estimates, you derive the expected time to complete an activity. The expected time is also referred to as the average time for the activity.

To calculate the expected time for each activity, you use the following equation:

$$T_e = \frac{T_0 + 4T_m + T_p}{6}$$

Where T_e is the expected time, T_0 is the optimistic time, T_m is the most likely time, and T_p is the pessimistic time required to complete an activity.

According to this equation, you calculate the expected time for an activity as a weighted mean of the optimistic, most likely, and pessimistic times. The weights attached to these times are 1, 4, and 1, respectively.
Consider the example of the activities in a software project as given in Table 1. The time estimates provided in Table 1 are the estimated times for each activity. Table 2 provides the break-up of the estimated times into the optimistic, pessimistic, and most likely times for each activity.

Table 2: Optimistic, Most Likely Time, and Pessimistic Estimates for Activities

Tasks	Optimistic Time Estimates (person days)	Most Likely Time Estimates (person days)	Pessimistic Time Estimates (person days)
Requirement analysis and project planning	7	10	13
Setting up the environment	3	6	9
Software construction	48	83	100
Unit testing	20	28	33
User training	4	5	6
System testing	10	15	20
User documentation	23	28	45
Data migration	18	18	30
Conducting user acceptance test	14	21	22

Using the formula for calculating estimated time for an activity, the estimated time for requirements analysis and project planning is:

$$T_e = \frac{7 + 4(10) + 13}{6} = 60 / 6 = 10 \quad persondays$$

Similarly, you can calculate the estimated times for all the activities in Table 2 using the above formula.

Figure 12 shows the PERT network schedule created using the time estimates.

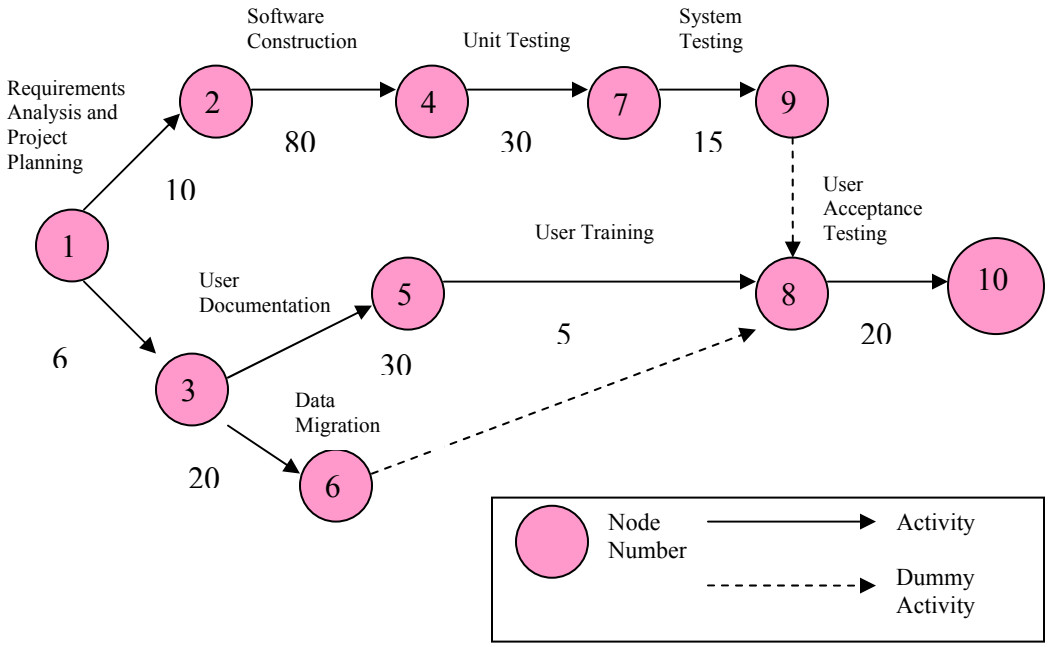


Figure 12: PERT Network Schedule for Activities

ii. PERT packages and enhancements

Some enhanced versions of the PERT chart support additional planning activities, such as personnel assignment, resource allocation and cost analysis. The chart can then draw attention to situations where personnel are assigned more responsibilities than they can handle, or where allocation of resources conflicts.

An interesting adaptation of PERT, called flow graph representation, which was developed by Riggs and Jones (1990), uses precedence networks to perform project life cycle cost analysis. The flow graph technique analyzes project costs based on relationships between quantities, unit cost, time variables, staffing costs and learning etc., all of which are represented on the PERT- like chart.

The flow graph representation technique places a significant amount of information on the network graph. This information, just like the basic PERT information, must be kept constantly updated. A small change to a large PERT chart can require the complete redrawing of the chart and the recalculation of the critical path. The resulting tedium does not promote much enthusiasm for

keeping the chart updated. For this reason, many computerized PERT utilities have been developed.

PERT software packages have been available for many years, but it is only during the past few years that good professional PERT packages have become available on PCs and other small computers. These packages take much of the tedium out of the preparation of PERT charts, and also come with additional features such as various planning analyzers for activity assignment, what if scenarios and resource allocation.

Computer utilities have been developed to perform flow graph representation analysis which produces scheduled costs for major project activities2. These utilities have proven invaluable for project managers and release managers from laborious desk work, providing them with more time to actively manage the project.

LECTURE # 39

9. Risk and Change Management

9.1 Risk is defined as the possibility of loss. It is the inability to achieve program objectives within defined cost, schedule, and technical constraints. Risk management is a set of actions that helps the project manager plan an approach to deal with uncertain occurrences.

A software project encounters two types of risks, development process risks and product- related risks. Some of the development process risks are developer errors, natural disasters, disgruntled employees, and poor management objectives. Some project related risks are incomplete requirements, unclear project deliverables and objectives, and complexity of the product.

The steps of risk management involve risk identification, risk analysis, and risk mitigation. Risk identification involves identifying risks. Risks are identified after discussion with team members about the requirements documents, available technology, resources, and other project-related factors.

Contingency planning involves maintaining an alternative plan if the original plan fails. Contingency plans are put to use when risks become a reality.

What is it? Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty. Many problems can plague a software project A risk is a potential problem - it might happen, it might not But regardless of the outcome, it's a really good idea to identify it, assess its probability of occurrence, estimate its impact, and establish a contingency plan should the problem actually occur.

Who does it? Everyone involved in the software process – managers, software engineers and customers - participate in risk analysis and management
Why is it important? Think about the Boy Scout motto: "Be prepared" Software is a difficult undertaking. Lots of things can go wrong, and frankly, many often do. It's for this reason that being prepared, understanding the risks and taking preventive measures to avoid or manage them is a key element of good software project management.

What are the steps? Recognizing what can go wrong is the first step, called - risk identification. Next, each risk is analyzed to determine the likelihood that it will occur and the damage that it will do if it does occur. Once this information is established, risks are ranked by probability and impact. Finally, a plan is developed to manage those risks with high probability and high impact.

What is the work product? A risk mitigation monitoring and management (RMMM) plan or a set of risk information sheets is produced. How do I ensure that I've done it right? The risks that are analyzed and managed should be derived from thorough study of the people, the product, the process and the project. The RMMM should be revisited as the project proceeds to ensure that risks are kept up to date. Contingency plans for risk management should be realistic.

9.2 Risk Management Concepts

Ever wondered why people insure their lives, their homes, their cars, or their valuables? Suppose your house, along with all the valuables, is burgled. You will be definitely disheartened. However, you will suffer less if all your valuables are insured. At times such as these, you realize the importance of planning in advance for the uncertain events in your life. Just as you plan for unforeseen events in your life, project managers also need to prepare for uncertainties affecting projects. All project management skills of a project manager can fall in the face of uncertainties and unplanned problems.

Robert Charette [CHA89] presents a conceptual definition of risk:

- First, risk concerns future happenings, Today and yesterday are beyond active concern, as we are already reaping what was previously sowed by our past actions, The question is, can we, therefore, by changing our actions today, create an opportunity for a different and hopefully better situation for ourselves tomorrow

- This means, second, that risk involves change, such as in changes of mind, opinion, actions, or places

- Third risk involves choice, and the uncertainty that choice itself entails, thus paradoxically, risk, like death and taxes, is one of the few certainties of life,

When risk is considered in the context of software engineering, Charette's three conceptual underpinnings are always in evidence. The future is our concern – what risks might cause the software project to go awry? Change is our concern - how will changes in customer requirements, development technologies, target computers, and all other entities connected to the project affect timeliness and overall success? Last, we must grapple with choices - what methods and tools should we use, how many people should be involved, how much emphasis on quality is "enough"?

Any project can encounter uncertainties in the form of increased costs, schedule delays, and diminished qualities. Unless tackled, these uncertainties can lead to major project disasters. The uncertainties encountered during project execution are the potential project risks. Every software project has to grapple with the new risks threatening information

security along with the conventional risks, such as hardware failure, time and cost escalation, defects, or resource crunch. Risk can be defined as the possibility of loss. Risk arises due to the inability to achieve objectives within defined cost, schedule, and technical constraints. Risk has two components. The possibility of not achieving a particular outcome is one, and the result of failing to achieve the outcome is the other. The former is the probability of loss, and the latter is the loss. Software project management deals with managing both these components of risk.

According to the risk management guru Barry Boehm, "Risk management focuses the project manager's attention on those portions of the project most likely to cause trouble and compromise participants' win conditions." In other words, risk management is a set of actions that helps the project manager plan to deal with uncertain occurrences. It is through risk management that project managers assess risks and manage to reduce risks to an acceptable level.

Software risk management is not about managing risks faced by a software organization. Here, the focus is on managing risks encountered during software development process. Therefore, the concern is about managing the future of a software project.

In this chapter, you will 1ook at the unforeseen events that might affect a software project. You will also learn about the steps for managing and mitigating software project risks.

Types of Risks

To be able to manage project risks, you must first understand what constitutes, a risk. All uncertain occurrences are not risks. Only those occurrences that have an adverse impact on the progress of a project are risks to the project. Risk is not a bad thing. Risk is bad only when it results in loss for an organization. Unless there is a potential for loss, there is no risk. Moreover, loss can be interpreted as either a bad outcome or a lost opportunity. The tendency of most project managers is to jump at the statement this is a risk. However, the desired reaction is to pre-empt all possible outcome and plan for them. Project risks can be broadly categorized into development process risks and product risks.

Development Process Risks

The risks encountered during product development are categorized as development process risks. These comprise developer errors, natural disasters, disgruntled employees, and poor management objectives.

Developer errors could be attributed to poor training due to budgetary constraints and inadequate skills and software tools. Ergonomic problems, environment problems, and interruptions or distractions at office also account for developer risks. Other risks in this category include problems in personnel acquisition and retention.

Similarly, natural disasters such as flood, cyclone, fire, storm, and snowfall are also risks to a project.

Disgruntled employees can also become a risk to an organization. For example, a sacked employee can use password snuffers to gain unauthorized access. A dismissed person can

flood the system with senseless messages. A disgruntled employee can also try to sabotage the project work by destroying files and programs.

A poorly defined management objective is another development process risk. If the language in the management objective is ambiguous and not stated clearly, the risk management program will not function properly. Narrowly focused and changing objectives that are not updated can also be counted as risks. Lack of contingency plans, incomplete cost estimates, and unrealistic schedules are also potential risks in a project. Similarly, unrealistic performance standards are also potential risks to the development process. Other possible risks include contractual risks, technological risks, and inadequate documentation of other concurrent projects.

Product Risks

Product risks crop up in the form of changing requirements during product development. Incomplete and unclear requirements are a risk to the product during development. Similarly, problems in meeting design specifications can also be categorized as risk to product development.

Risks could arise if the project deliverables or objectives are not clearly defined or if technical data is missing. The possibility of several alternatives at any given time during the project is also a cause of concern. If errors are not recognized during the design phase, they could turn into risks for the project. Similarly, risks could arise due to the size and complexity of the product or while achieving client acceptance of the product.

> **Note:**
> The key idea in risk management is not to wait for a risk to materialize and become a problem. The objective of risk management is to ensure that for each perceived risk, you know well in advance how to tackle it.

Risk Management Process

Project Risk Management includes the processes concerned with conducting risk management planning, identification, analysis, responses, and control on a project. The objectives of Risk Management are to increase the probability and impacts of positive events and decrease the probability and impacts of events adverse to project objectives.

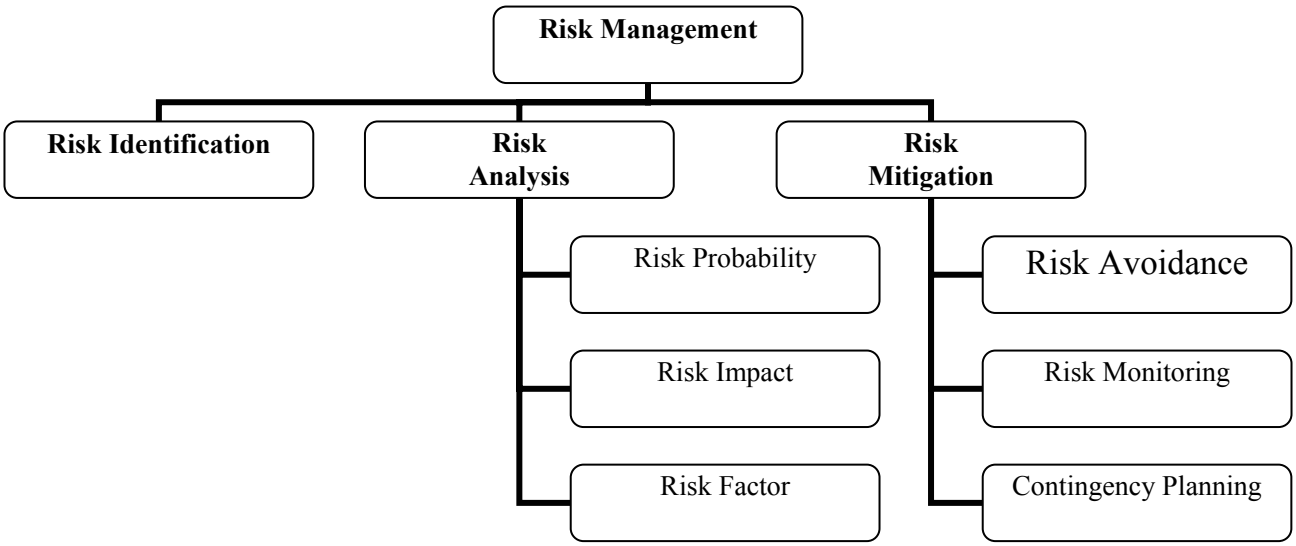
The process of risk management begins during the analysis phase of software development life cycle. However, the actual process of managing risks continues throughout the product development phase. Risk management is a dynamic process because it deals with the activities that are yet to happen. Risk management has a two-fold agenda. First, deciding actions for preventing risks from happening, and second, deciding actions for tackling risks that materialize. Therefore, risk management is all about pre-empting a risk, coming up with a plan for resolving the risk, and finally executing the plan.

Figure 6.1 displays the steps of the risk management process. Formally, articulated, risk management process consists of three steps:

1. Risk identification
2. Risk analysis

3. Risk mitigation

Figure 6.1: Risk Management Process



Risk Identification

In this step, the project manager gathers information about the potential risks in the project. The project manager plans the strategies for avoiding risks or controlling them. The project team conducts brainstorming sessions and discussions among team members about the requirements document. They discuss the available technology, manpower, prevailing environment, and all project-related factors. The project manager picks up the thread from these and creates a risk log. After the risk log is prepared, the project manager calls a meeting within the team and technical experts to discuss the risk log and the mitigation plans. An effective way of identifying' risks is using a questionnaire.

Table 6.1 displays a sample risk identification questionnaire.

Table 6.1: *Sample Risk Identification Questionnaire*

SN	Risk Description	Yes/No/Not Applicable (NA)
A.	**Product Engineering**	
1)	Are requirements changing continuously during product development?	
2)	Do the changing requirements affect each of the following:	
	Quality	
	Functionality	
	Schedule	
	Integration	
	Design	
	Testing	

3)	Are the external interfaces changing?	
4)	Are the requirements missing or incompletely specified?	
5)	Are there any missing requirements?	
6)	Can these requirements be incorporated into the system?	
7)	Does the client have unwritten requirements or expectations?	
8)	Is there a way to capture these requirements?	
9)	Are requirements unclear or in need of interpretation?	
10)	Are you able to understand the requirements as written?	
11)	Will the requirements lead to the product the client wants?	
12)	Are there any requirements that may not specify what the client really wants?	
B.	**Product Design**	
1)	Do you encounter problem in meeting functionality requirements?	
2)	Are there any specified algorithms that may not satisfy the requirements?	
3)	Will the design and/or implementation be difficult to achieve?	
4)	Are there any requirements or functions that are difficult to design?	
C.	**Reusability**	
1)	Are you reusing the program?	
2)	Do you foresee any problems in documentations?	
3)	Do you foresee any problems in performance?	
4)	Do you foresee any problems in functionality?	

The sample questionnaire in Table 6.1 includes an exhaustive list of risks that might be encountered during the progress of a project. The answers to the questions in the risk identification questionnaire enable the project manager to estimate the impact of risks. In the sample questionnaire Table 6.1, the risks are categorized under product engineering, product-design, and reusability. From this table, you obtain a list of risks that are relevant to each category. You compare the information obtained from this table with past results and estimate the criticality of risk. In short, during risk identification, you obtain answers to the following queries:

- **Why** is the risk important?
- **What** information is needed to track the status of the risk?
- **Who** is responsible for the risk management activity?
- **What** resources are needed to perform the activity?
- What is the **detailed plan** to improve the risk and / or mitigate it?

Risk Analysis

After identifying the risks, the project manager needs to analyze the risks. Uncertainty and loss are the two characteristics of risk. The uncertainty factor in risk means that the unknown event mayor may not happen. While analyzing risks, the project manager needs to quantify the level of uncertainty and the degree of loss. Based on this, the project manager plans schedules and costs. During analysis, information on risk is converted into information on decision-making. Analysis provides the basis for the project manager to work on the "right" risks.

Table 6.2: *Risk Analysis Table*

Risk Description	Probability of Occurrence (0 – 1)	Impact on Project (1 – 10)	Risk Factor (Probability x Impact)

Table 6.2 displays a risk analysis format. There are various tasks involved in risk analysis. First, the WBS elements are identified. One of the tasks in the risk analysis phase is to describe the risk. The risk can be product-related, process-related, organization-related, client-related, or infrastructure-related.

Second, the WBS elements are evaluated to determine the risk events. Then the project manager quantifies the probability of occurrence of risk. The project manager can assign probability values between 0 and 1. For example, a risk with a low probability of occurrence is marked 0.2 while that with a high probability of occurrence is marked 0.8. The reason why a particular risk has a high or low probability depends on the actual circumstance of the project.

Third, the risks are rated depending on their probability of occurrence. Based on the probability of risk, the project manager identifies the impact of the risk. The impact of risk on cost, schedule, and quantity needs to be calculated and graded. The impact of risk can be graded on a scale of 1 to 10, 1 being the lowest, and 10 being the highest.

Then the risk factor is calculated by multiplying the probability of risk and the impact of risk. Finally, each risk is prioritized relative to other risks. The risk factor is used to prioritize the identified risks. For example, the risk with a probability value 0.1 and an impact value 2 will have minimal impact. While risks close to probability value 0.8 and with an impact value 9 will have greater impact. Therefore, the project manager can prioritize risks based on the probability and the impact of risks. A risk that has a high impact and low probability will not absorb a significant amount of the project manager's time. However, high-impact risks with moderate to high probability will catch the attention of the project manager.

Risk Mitigation

Risk mitigation is the best possible approach adopted by the project manager to avoid risks from occurring. The probability of the risk occurring and the potential impact of the risk can be mitigated by dealing with the problem early in the project. Essentially, risk mitigation involves three possibilities and the project manager needs to adopt a risk mitigation strategy aimed at them. The three possibilities include:

- Risk avoidance
- Risk monitoring
- Contingency planning

Risk Avoidance

To avoid risks from occurring, the project team prepares the risk plan before the commencement of the project. The project team identifies the potential risks and prioritizes them based on their probability of occurrence and impact. Then, the team prepares a plan for managing risks. In most software projects, this plan is popularly called the risk management plan. Table 6.3 displays the format of a risk management plan.

Table 6.3: *Risk Management Plan*

Risk Description	Probability of Occurrence (0 – 1)	Impact on Project (1 – 10)	Risk Factor (Probability x Impact)	Mitigation Steps	Responsibility	Start Date	End Date

To prepare the risk management plan, the project team first identifies and assesses the risks associated with the project. Then, the probability of occurrence of each risk is estimated and the possible impact is calculated. In the plan, the probable cost and damage is also quantified. The project team also identifies the contingency plans for all the identified risks. The contingency plan for each risk is based on the project's defined operational software process. The plan is modified throughout the software development life cycle of the project based on the changes taking place. The contingency plan also includes the cost, in terms of effort, in carrying out the plan. The software risks identified are tracked, reassessed, and re-planned at the end of each phase. The project manager revisits the plan if significant changes are introduced in the software project.

Risk Monitoring

As the project proceeds, risk-monitoring activities commence. It is not possible to monitor closely all the risks that are identified for the project. For example, if 100 risks are identified for a project, only top 20 risks are monitored. There are re-planning checkpoints where the information obtained from monitoring the risks is used to refine the risk assessments and management plan. The project manager monitors the top 20

percent of the factors that may indicate the status of the risks in the project. In the case of large teams, the project manager also needs to monitor the attitude of the team members and their problems. This helps the project manager monitor any possible team-related risks.
Besides monitoring the top 20 percent of the risks, the project manager needs to monitor the mitigation steps also. Consider an example. To ensure that particular software is browser- independent, the software is created on the lowest compatible browser. Such software will work on any browser thus making it browser-independent. Therefore, mitigating a project risk involves working hard at reducing the possibility that the risk will ever occur. Mitigation includes nearly all actions that a project team takes to overcome risks. For example, choosing a more expensive but proven technology over, a newer, less expensive technology is a step toward mitigating project risks.

Contingency Planning
The possibility of contingency planning arises when mitigation efforts fail and risk becomes a reality. Contingency planning is used to monitor risks and invoke a predetermined response. According to the plan, a trigger is set up. If the trigger is reached, the contingency plan is put into effect. Contingency planning involves maintaining an alternative plan if the original plan fails. A simple example could be the savings people make for a rainy day. Contingency plans are a must for the top 20 percent of the risks identified. These plans are put to use after the risks become a reality. The importance of contingency planning can be realized from this example. Despite the massive attack on WTC, the stock markets in the US resumed functioning within a few days. This was possible because the finance companies had backed up their data and information on computers elsewhere. The contingency planning of finance companies prevented the risk of huge data loss for the stock market.

Managing Risks: Case Study
Consider a scenario. Your organization is a vendor of software solutions. A bus transport company the US wants you to develop a Schedule Adherence system. The team that will develop this software is new and the platform selected for development is also new to your organization. The project team needs to be trained intensively for this.

During this project, the team is expected to manage a large volume of data. The team has never had any experience in managing such a large volume of data. The system also needs to use this data to generate various MIS reports related to delays or adherence of bus services.

The performance requirement is less than fifteen seconds for all popular browsers. Your organization is anticipating numerous requirement changes during the development process. The system needs to be implemented across several states in the country. The data related to the system is highly confidential because it can provide an edge to the competitors.

Now, as a project manager, you need to prepare a risk management plan for this project. The project starts on May 15 and should be completed on November 15.

First, you need to identify the potential risks involved in the project. The potential risks to the project are described in Table 6.4.

Table 6.4: *Potential Risks to a Project*

Risk Description
Inexperienced staff
Performance risk due to high volume of data to be processed
Cross-browser compatibility
Involvement of new technology
Design changes during development

After identifying the risks, you need to estimate the probability of their occurrence and their impact on product development. Based on this, you calculate the risk factor and plan the mitigation steps.

Your risk management plan is displayed in Table 6.5.

Table 6.5: *The Risk Management Plan for Building a Schedule Adherence System*

Risk Description	Probability of Occurrence (0 – 1)	Impact on Project (1 – 10)	Risk Factor (Probability x Impact)	Mitigation Steps	Responsibility	Start Date	End Date
Inexperience Staff	0.8	3	2.4	Project Manager	Conducting training sessions before the need commencement of project	May 15	July 15
Performance risk due to high volume of data to be processed	0.6	7	4.2	Architect	Massive tuning of architecture during the design phase and conducting a proof of concepts for the design	May 15	Till the end of the project
Cross-browser	0.6	5	3.0	Developer	Using the lowest	May 15	Till the

compatibility					compatible browser for development		end of the project
Involvement of new technology	0.5	5	2.5	Project Manager	Ensuring all details pertaining to the technology is available and keeping in close touch with the technology vendor	May	Till the end of the project
Design changes during development	0.6	8	4.8	Architect	Designing a flexible architecture that can accommodate future changes and enhancement	May 15	Till the end of the project

In Table 6.5, the risk factors with high values are the high priority risks. Here, the high priority risks are design changes during development, performance risk due to high volume of data to be processed, and cross-browser compatibility. The high priority risks need to be monitored closely and continuously. The rest of the risks can be monitored periodically. Certain risks need contingency planning despite being low in the priority list. For example, the risk due to involvement of new technology is a low priority risk. However, the probability of occurrence of this risk is 0.5, which is fairly high. This calls for contingency planning because you may not be aware of all the details of the new technology. Suppose the system fails to respond if the number of users exceeds a certain number. In such a situation, you need to have a contingency plan ready. You need to discuss with the vendor regarding a workaround for such a situation. The workaround should be ready in the case of failure of the system.

Robert Charette [CHA89] presents a conceptual definition of risk:

First, risk concerns future happenings, Today and yesterday are beyond active concern, as we are already reaping what was previously sowed by our past actions, The question is, can we, therefore, by changing our actions today, create an opportunity for a different and hopefully better situation for ourselves tomorrow, This means, second, that risk involves change, such as in changes of mind, opinion, actions, or places…[Third] risk involves choice, and the uncertainty that choice itself entails, Thus paradoxically, risk, like death and taxes, is one of the few certainties of life,

When risk is considered in the context of software engineering, Charette's three conceptual underpinnings are always in evidence. The future is our concern – what risks might cause the software project to go awry? Change is our concern -how will changes in customer requirements, development technologies, target computers, and all other entities connected to the project affect timeliness and overall success? Last, we must grapple with choices - what methods and tools should we use, how many people should be involved, how much emphasis on quality is "enough"?

What is it? Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty. Many problems can plague a software project A risk is a potential problem - it might happen, it might not But regardless of the outcome, it's a really good idea to identify it, assess its probability of occurrence, estimate its impact, and establish a contingency plan should the problem actually occur.
Who does it? Everyone involved in the software process – managers, software engineers and customers - participate in risk analysis and management
Why is it important? Think about the Boy Scout motto: "Be prepared" Software is a difficult undertaking. Lots of things can go wrong, and frankly, many often do. It's for this reason that being prepared, understanding the risks and taking preventive measures to avoid or manage them is a key element of good software project management.
What are the steps? Recognizing what can go wrong is the first step, called - risk identification. Next, each risk is analyzed to determine the likelihood that it will occur and the damage that it will do if it does occur. Once this information is established, risks are ranked by probability and impact. Finally, a plan is developed to manage those risks with high probability and high impact.
What is the work product? A risk mitigation monitoring and management (RMMM) plan or a set of risk information sheets is produced.
How do I ensure that I've done it right? The risks that are analyzed and managed should be derived from thorough study of the people, the product, the process and the project. The RMMM should be revisited as the project proceeds to ensure that risks are kept up to date. Contingency plans for risk management should be realistic.

6.1 REACTIVE VS. PROACTIVE RISK STRATEGIES

Reactive strategies have been laughingly called the “Indiana Jones School of risk management” [THO92]. In the movies that carried his name, Indiana Jones, when faced with overwhelming difficulty, would invariably say, “Don’t worry, I’ll think of something!” Never worrying about problems until they happened, Indy would react in some heroic way.
Sadly, the average software project manager is not Indiana Jones and the members of the software project team are not his trusty sidekicks. Yet, the majority of software teams rely solely on reactive risk strategies. At best, a reactive strategy monitors the project for likely risks. Resources are set aside to deal with them, should they become actual problems. More commonly, the software team does nothing about risks until something goes wrong. Then, the team flies into action in an attempt to correct the problem rapidly.

This is often called a fire fighting mode. When this fails, “Crisis Management” [CHA92] takes over, and the project is in real jeopardy.

A considerably more intelligent strategy for risk management is to be proactive. A proactive strategy begins long before technical work is initiated. Potential risks are identified, their probability and impact are assessed and they are ranked by importance. Then, the software team establishes a plan for managing risk. The primary objective is to avoid risk, but because not all risks can be avoided, the team works to develop a contingency plan that will enable it to respond in a controlled and effective manner.

6.2 SOFTWARE RISKS

Although there has been considerable debate about the proper definition for software risk, there is general agreement that risk always involves two characteristics [HIG 95]:

- Uncertainty - the risk may or may not happen; that is, there are no 100 % probable risks.
- Loss - if the risk becomes a reality, unwanted consequences or losses will occur.

When risks are analyzed, it is important to quantify the level of uncertainty and the degree of loss associated with each risk. To accomplish this, different categories of risks are considered.

Project risks threaten the project plan. That is, if project risks become real, it is likely that project schedule will slip and that costs will increase. Project risks identify potential budgetary, schedule, personnel (staffing and organization), resource, customer, and requirements problems and their impact on a software project. In Chapter 5, project complexity, size, and the degree of structural uncertainty were also defined as project (and estimation) risk factors.

Technical risks threaten the quality and timeliness of the software to be produced. If a technical-risk becomes a reality, implementation may become difficult or impossible. Technical risks identify potential design, implementation, interface, verification, and maintenance problems. In addition, specification ambiguity, technical uncertainty, technical obsolescence, and "leading-edge" technology are also risk factors. Technical risks occur because the problem is harder to solve than we thought it would be.

Business risks threaten the viability of the software to be built. Business risks often jeopardize the project or the product. Candidates for the top five business risks are (1) building a excellent product or system that no one really wants (market risk), (2) building a product that no longer fits into the overall business strategy for the company (strategic risk), (3) building a product that the sales force doesn't understand how to sell, (4) losing the support of senior management due to a change in focus or a change in people (management risk), and (5) losing budgetary or personnel commitment (budget risks). It is extremely important to note that simple categorization won't always work. Some risks are simply unpredictable in advance.

Another general categorization of risks has been proposed by Charette [CHA89J. Known risks are those that can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is being developed, and other reliable information sources (e.g., unrealistic delivery date, lack of documented requirements or software scope, poor development environment). Predictable risks are extrapolated from past project experience (e.g., staff turnover, poor communication with the customer, dilution of staff effort as ongoing maintenance requests are serviced). Unpredictable risks are the joker in the deck. They can and do occur, but they are extremely difficult to identify in advance.

6.3 RISK IDENTIFICATION

Risk identification is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.). By identifying known and predictable risks, the project manager takes a first step toward avoiding them when possible and con- trolling them when necessary.

There are two distinct types of risks for each of the categories that have been presented in Section 6.2: generic risks and product-specific risks. Generic risks are a potential threat to every software project. Product-specific risks can be identified only by those with a clear understanding of the technology, the people, and the environment that is specific to the project at hand. To identify product -specific risks, the project plan and the software statement of scope are examined and an answer to the following question is developed: What special characteristics of this product may threaten our project plan?

One method for identifying risks is to create a risk item checklist. The checklist can be used for risk identification and focuses on some subset of known and predictable risks in the following generic subcategories:

- Product size-risks associated with the overall size of the software to be built or modified.
- Business impact-risks associated with constraints imposed by management or the marketplace.
- Customer characteristics-risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.
- Process definition-risks associated with the degree to which the software process has been defined and is followed by the development organization.
- Development environment-risks associated with the availability and quality of the tools to be used to build the product.
- Technology to be built-risks associated with the complexity of the system to be built and the newness of the technology that is packaged by the system.
- Staff size and experience-risks associated with the overall technical and project experience of the software engineers who will do the work.

The risk item checklist can be organized in different ways. Questions relevant to each of the topics can be answered for each software project. The answers to these questions allow the planner to estimate the impact of risk. A different risk item checklist format simply lists characteristics that are relevant to each generic subcategory. Finally, a set of "risk components and drivers" [AFC88] are listed along with their probability of occurrence. Drivers for performance, support, cost, and schedule are discussed in answer to later questions.
A number of comprehensive checklists for software project risk have been proposed in the literature (e.g., [SEI93], [KAR96]). These provide useful insight into generic risks for software projects and should be used whenever risk analysis and management is instituted. However, a relatively short list of questions [KEI98] can be used to provide a preliminary indication of whether a project is "at risk."

6.3.1 Assessing Overall Project Risk

The following questions have derived from risk data obtained by surveying experienced software project managers in different part of the world [KEI98]. The questions are ordered by their relative importance to the success of a project.

1. Have top software and customer managers formally committed to support the project?
2. Are end-users enthusiastically committed to the project and the system/product to be built?
3. Are requirements fully understood by the software engineering team and their customers?
4. Have customers been involved fully1n the definition of requirements?
5. Do end-users have realistic expectations?
6. Is project scope stable?
7. Does the software engineering team have the right mix of skills?
8. Are project requirements stable?
9. Does the project team have experience with the technology to be implemented?
10. Is the number of people on the project team adequate to do the job?
11. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

If anyone of these questions is answered negatively, mitigation, monitoring, and /management steps should be instituted without fail. The degree to which the project is at risk is directly proportional to the number of negative responses to these questions.

6.3.2 Risk Components and Drivers

The U.S. Air Force [AFC88j has written a pamphlet that contains excellent guidelines (or software risk 1dentification and abatement. The Air Force approach requires that the project manager identify the risk drivers that affect software risk components-
Performance, cost, support, and schedule. In the context of this discussion, the risk components are defined in the following manner:

- Performance risk - the degree of uncertainty that the product will meet its requirements and be fit for its intended use.
- Cost risk - the degree of uncertainty that the project budget will be maintained.
- Support risk - the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
- Schedule risk - the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

The impact of each risk driver on the risk component is divided into one of four impacts categories-negligible, marginal, critical, or catastrophic. Referring to Figure 6.1 [BOE89], a characterization of the potential consequences of errors (rows labeled 1) or a failure to achieve a desired outcome (rows labeled 2) are described. The impact category is chosen based on the characterization that best fits the description in the table.

Figure 6.1 Impact assessment

Components/Category		Performance	Support	Cost	Schedule
Catastrophic	1	Failure to meet the requirement would result in mission failure		Failure results in increased costs and schedule delays with expected values of $500 K	
	2	Significant degradation to non achievement of technical performance	Non responsive or unsupportable software	Significant financial shortages, budget overrun likely	Unavoidable IOC
Critical	1	Failure to meet the requirement would degrade system performance to a point where mission success is questionable		Failure results in operational delays and/or increased costs with expected $100 K to $ 500 K	
	2	Some reduction in technical performance	Minor delays in software modification	Some shortage of financial resources, possible overruns	Possible slippage in IOC
Marginal	1	Failure to meet the requirement would result in degradation of secondary mission		Costs, impacts and / or recoverable schedule slips with expected value of $ 1 K to $ 100 K	

	2	Minimal to small reduction in technical performance	Responsive software support	Sufficient financial resources	Realistic achievable schedule
Negligible	1	Failure to meet the requirement would create inconvenience or non operational impact		Error results in minor cost and / or schedule impact with expected value of less than $ 1 K	
	2	No reduction in technical performance	Easily supportable software	Possible budget under run	Early achievable IOC

Note:

(1) The potential consequence of undetected software errors or faults.

(2) The potential consequence if the desired outcome is not achieved.

6.4 RISK PROJECTION

Risk projection, also called risk estimation, attempts to rate each risk in two ways-the likelihood or probability that the risk is real and the consequences of the problems associated with the risk, should it occur. The project planner, along with other managers and technical staff, performs four risk projection activities:

(1) Establish a scale that reflects the perceived likelihood of a risk,
(2) Delineate the consequences of the risk,
(3) Estimate the impact of the risk on the project and the product, and
(4) Note the overall accuracy of the risk projection so that there will be no misunderstandings.

6.4.1 Developing a Risk Table

A risk table provides a project manager with a simple technique for risk projection. A sample risk table is illustrated in figure 6.2.

Figure 6.2 Sample risk table prior to sorting

Risks	Category	Probability	Impact	RMMM
Size estimate may be significantly low	PS	60%	2	
Larger number of users than planned	PS	30%	3	
Less reuse than planned	PS	70%	2	
End-users resist system	BU	40%	3	
Delivery deadline will be lightened	BU	50%	2	
Funding will be lost	CU	40%	1	
Customer will change requirements	PS	80%	2	
Technology will not meet expectations	TE	30%	1	
Lack of training on tools	DE	80%	3	

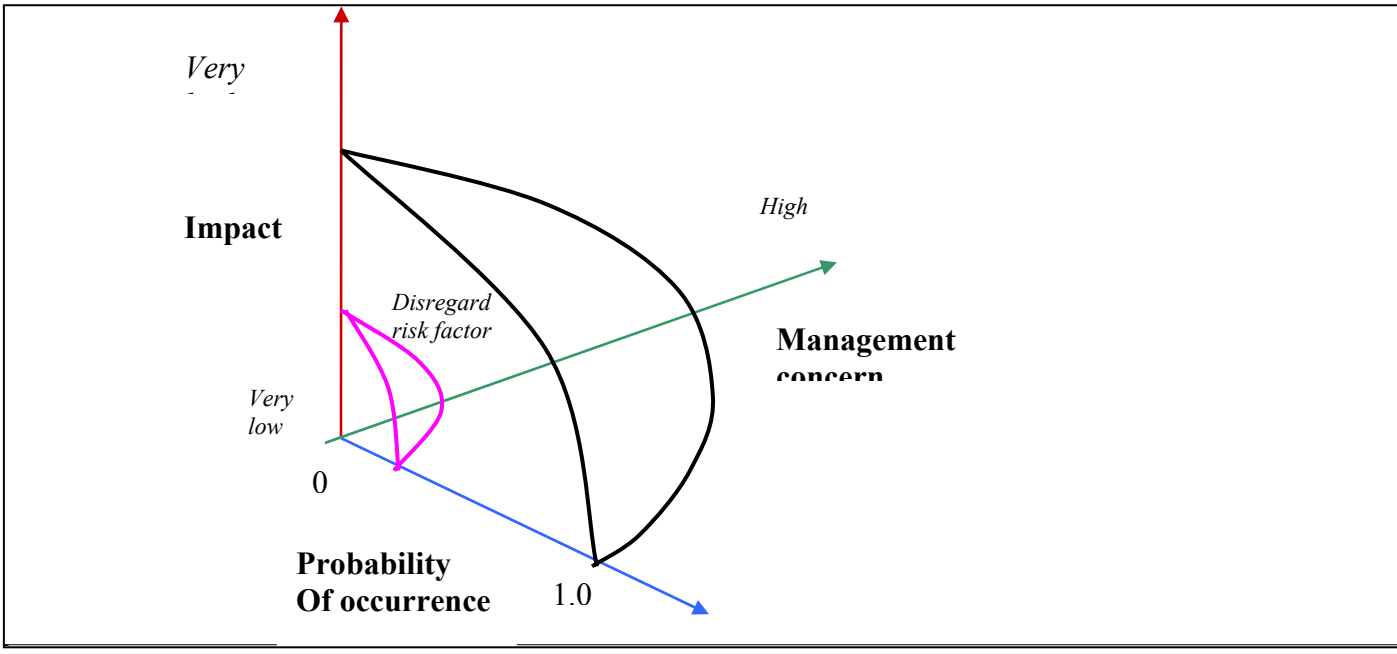
Staff inexperienced Staff turnover will be high	ST ST	80% 60%	2 2	

Impact values:
l- Catastrophic
2- Critical
3- marginal
4- Negligible

A project team begins by listing all risks (no matter how remote) in the first column of the table. This can be accomplished with the help of the risk item check-lists referenced in Section 6.3. Each risk is categorized in the second column (e.g., PS implies a project size risk, BU implies a business risk). The probability of occurrence of each risk is entered in the next column of the table. The probability value for each risk can be estimated by team members individually. Individual team members are polled in round-robin fashion until their assessment of risk probability begins to converge.

Next, the impact of each risk is assessed. Each risk component is assessed using the characterization presented in Figure 6.1, and an impact category is determined. The categories for each of .the four risk components - performance, support, cost, and schedule - are averaged to determine an overall impact value.

Figure 6.3 Risk and management concern



Once the first four columns of the risk table have been completed, the table is sorted by probability and by impact. High-probability, high-impact risks percolate to the top of the

table, and low-probability risks drop to the bottom. This accomplishes first order risk prioritization. The project manager studies the resultant sorted table and defines a cutoff line. The cutoff line (drawn horizontally at some point in the table) implies that only risks that lie above the line will be given further attention. Risks that fall below the line are re-evaluated to accomplish second-order prioritization. Referring to Figure 6.3, risk impact and probability have a distinct influence on management concern. A risk factor that has a high impact but a very low probability of occurrence should not absorb a significant amount of management time. However, high-impact risks with moderate to high probability and low-impact risks with high probability should be carried forward into the risk analysis steps that follow.

All risks that lie above the cutoff line must be managed. The column labeled RMMM contains a pointer into Risk Mitigation, Monitoring and Management Plan or alternatively, a collection of risk information sheets developed for all risks that lie above the cutoff. The RMMM plan and risk information sheets are discussed in Sections 6.5 and 6.6.

Risk probability can be determined by making individual estimates and then developing a single consensus value. Although that approach is workable, more sophisticated techniques for determining risk probability have been developed [AFC88]. Risk drivers can be assessed on a qualitative probability scale that has the following values: impossible, improbable, probable, and frequent. Mathematical probability can then be associated with each qualitative value (e.g., a probability of 0.7 to 1.0 implies a highly probable risk).

6.4.2 Assessing Risk Impact

Three factors affect the consequences that are likely if a risk does occur: its nature, its scope, and its timing. The nature of the risk indicates the problems that are likely if it occurs. For example, a poorly defined external interface to customer hardware (a technical risk) will preclude early design and testing and will likely lead to system integration problems late in a project. The scope of a risk combines the severity (just how serious is it?) with its overall distribution (how much of the project will be affected or how many customers are harmed?). Finally, the timing of a risk considers when and for how long the impact will be felt. In most cases, a project manager might want the "bad news" to occur as soon as possible, but in some cases, the longer the delay, the better.

Returning once more to the risk analysis approach proposed by the U.S. Air Force fAFC88), the following steps are recommended to determine the overall consequences of a risk:

1. Determine the average probability of occurrence value for each risk component.
2. Using Figure 6.1, determine the impact for each component based on the criteria shown.
3. Complete the risk table and analyze the results as described in the preceding sections. .

The overall risk exposure, RE, is determined using the following relationship [HAL98]:
RE=P x C.

where P is the probability of occurrence for a risk, and C is the cost to the project should the risk occur.
For example, assume that the software team defines a project risk in the following manner:

Risk identification. Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.

Risk probability. 80% (likely).

Risk impact. 60 reusable software components were planned. If only 70 percent can be used, 18 components would have to be developed from scratch (in addition to other custom software that has been scheduled for development). Since the average component is 100 LOC and local data indicate that the software engineering cost for each LOC is $14.00, the overall cost (impact) to develop the components would be 18 x 100 x 14 = $25,200.

Risk exposure. RE = 0.80 x 25,200 - $20,200.
Risk exposure can be computed for each risk in the risk table, once an estimate of the cost of the risk is made. The total risk exposure for all risks (above the cutoff in the risk table) can provide a means for adjusting the final cost estimate for a project. It can also be used to predict the probable Increase In staff resources required at various points during the project schedule.
The risk projection and analysis techniques described in Sections 6.4.1 and 6.4.2 are applied iteratively as the software project proceeds. The project team should revisit the risk table at regular intervals, re-evaluating each risk to determine when new circumstances cause its probability and impact to change. As a consequence of this activity, it may be necessary to add new risks to the table, remove some risks that are no longer relevant, and change the relative positions of still others.

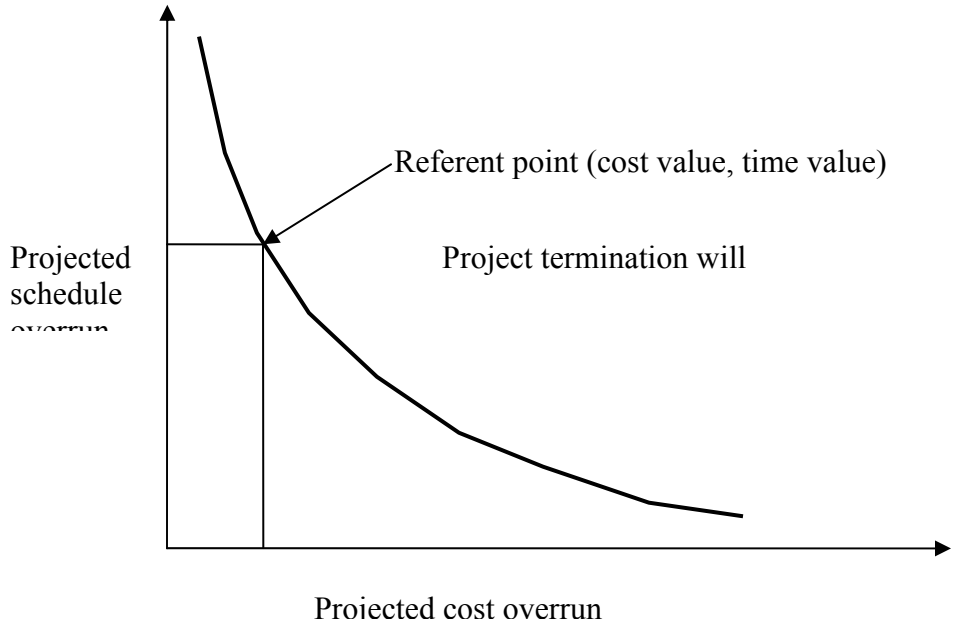
6.4.3 Risk Assessment

At this point in the risk management process, we have established a set of triplet of the form [CHA89]:

$[r_j, l_j, x_j]$

where r_j is risk, l_j is the likelihood (probability) of the risk, and x_j is the impact of the risk. During risk assessment, we further examine the accuracy of the estimates that were made during risk projection, attempt to rank the risks that have been uncovered, and begin thinking about ways to control and/or avert risks that are likely to occur.

Figure 6.4 Risk referent level



For assessment to be useful, a risk referent level [CHA89] must be defined. For most software projects, the risk components discussed earlier - performance, cost, support, and schedule also represent risk referent levels: That is, there is a level for performance, degradation, cost overrun, support difficulty, or schedule slippage (or any combination of the four) that will cause the project to be terminated. If a combination of risks create problems that cause one or more of these referent levels to be exceeded, work will stop. In the context of software risk analysis, a risk referent level has a single point, called the referent point or break point, at which the decision to proceed with the project or terminate it (problems are just too great) are equally weighted. Figure 6.4 represents this situation graphically.

In reality, the referent level can rarely be represented as a smooth line on a graph. In most cases it is a region in which there are areas of uncertainty; that is, attempting to predict a management decision based on the combination of referent values is often impossible. Therefore, during risk assessment, we perform the following steps:

1. Define the risk referent levels for the project.
2. Attempt to develop a relationship between each (r_j, l_j, x_j) and each of the referent levels.
3. Predict the set of referent points that define a region of termination, bounded by a curve or areas of uncertainty.
4. Try to predict how compound combinations of risks will affect a referent level.

6.5 RISK REFINEMENT

During early stages of project planning, a risk may be stated quite generally. As time passes and more is learned about the project and the risk, it may be possible to refine the risk into a set of more detailed risks, each somewhat easier to mitigate, monitor, and manage.

One way to do this is to represent the risk in condition-transition-consequence (CTC) II format [GLU94). That is, the risk is stated in the following form:

Given that <condition> then there is concern that (possibly) <consequence>.

Using the CTC format for the reuse risk noted in Section 6.4.2, we can write:
Given that all reusable software components must conform to specific design standards and that some do not conform, then there is concern that (possibly) only 70 percent of the planned reusable modules may actually be integrated into the as-built system, resulting in the need to custom engineer the remaining 30 percent of components. This general condition can be refined in the following manner:

Sub condition 1. Certain reusable components were developed by a third party with no knowledge of internal design standards.
Sub condition 2. The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.
Sub condition 3. Certain reusable components have been implemented in a language that is not supported on the target environment.

The consequences associated with these refined sub conditions remains the same (i.e., 30 percent of software components must be customer engineered), but the refinement helps to isolate the underlying risks and might lead to easier analysis and response.

6.6 RISK MITIGATION, MONITORING, AND MANAGEMENT

All of the risk analysis activities presented to this point have a single goal to assist the project team in developing a strategy for dealing with risk. An effective strategy must consider three issues:

- risk avoidance
- risk monitoring
- risk management and contingency planning
-

If a software team adopts a proactive approach to risk, avoidance is always the best strategy. This is achieved by developing a plan for risk mitigation. For example, assume that high staff turnover is noted as a project risk, r_1 Based on past history and management intuition, the likelihood, l_1 of high turnover is estimated to be 0.70 (70 per cent, rather high) and the impact, x_1 is projected at level 2. That is, high turnover will have a critical impact on project cost and schedule.

To mitigate this risk, project management must develop a strategy for reducing turnover. Among the possible steps to be taken are

- Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, and competitive job market).
- Mitigate those causes that are under our control before the project starts.
- Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.
- Organize project teams so that information about each development activity is widely dispersed.
- Define documentation standards and establish mechanisms to be sure that .documents are developed in a timely manner.
- Conduct peer reviews of all work (so that more than one person is “up to speed").
- Assign a backup staff member for every critical technologist.

As the project proceeds, risk monitoring activities commence. The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely. In the case of high staff turnover, the following factors can be monitored:

- General attitude of team members based on project pressures.
- The degree to which the team has jelled.
- Interpersonal relationships among team members.
- Potential problems with compensation and benefits.
- The availability of jobs within the company and outside it.

In addition to monitoring these factors, the project manager should monitor the effectiveness of risk mitigation steps. This is one mechanism for ensuring continuity, should a critical individual leave the project. The project manager should monitor documents carefully to ensure that each can stand on its own and that each imparts information that would be necessary if a newcomer were forced to join the software team somewhere in the middle of the project.

Risk management and contingency planning assumes that mitigation efforts have failed and that the risk has become a reality. Continuing the example, the project is well underway and a number of people announce that they will be leaving. If the mitigation strategy has been followed, backup is available, information is documented, and knowledge has been dispersed across the team. In addition, the project manager may temporarily refocus resources (and readjust the project schedule) to those functions that are fully staffed, enabling newcomers who must be added to the team to "get up to speed." Those individuals who are leaving are asked to stop all work and spend their last weeks in "knowledge transfer mode. This might include video-based knowledge capture, the development of "commentary documents," and/or meeting with other team members who will remain on the project.

It is important to note that RMMM steps incur additional project cost. For example, spending the time to "backup" every critical technologist costs money. Part of risk management, therefore, is to evaluate when the benefits accrued by the RMMM steps are outweighed by the costs associated with implementing them. In essence, the project

planner performs a classic cost/benefit analysis. If risk aversion steps for high turnover, it will increase both project cost and duration by an estimated 15 percent, but the predominant cost factor is "backup," management may decide not to implement this step. On the other hand, if the risk aversion steps are projected to' increase costs by 5 percent and duration by only 3 percent management will likely put all into place.

For a large project, 30 or 40 risks may be identified. If between three and seven risk management steps are identified for each, risk management may become a project in itself! For this reason, we adapt the Pareto 80-20 rule to software risk. Experience indicates that 80 percent of the overall project risk (i.e., 80 percent of the potential for project failure) can be accounted for by only 20 percent of the identified risks. The work performed during earlier risk analysis steps will help the planner to determine which of the risks reside in that 20 percent (e.g., risks that lead to the highest risk exposure. For this reason, some of the risks identified, assessed, and projected may not make it into the RMMM plan - they don't fall into the critical 20 percent (the risks with highest project priority).

6.7 SAFETY RISKS AND HAZARDS

Risk is not limited to the software project itself. Risks can occur after the software has been successfully developed and delivered to the customer. These risks are typically associated with the consequences of software failure in the field.

In the early days of computing, there was reluctance to use computers (and soft- ware) to control safety critical processes such as nuclear reactors, aircraft flight control, weapons systems, and large-scale industrial processes. Although the probability .of failure of a well-engineered system was small, an undetected fault in a computer- based control or monitoring system could result in enormous economic damage or, worse, significant human injury or loss of life. But the cost and functional benefits of Computer-based control and monitoring far outweigh the risk. Today, computer hardware and software are used regularly to control safety critical systems.

When software is used as part of a control system, complexity can increase by an order of magnitude or more. Subtle design faults induced by human error-some- thing that can be uncovered and eliminated in hardware-based conventional control-become much more difficult to uncover when software is used.

Software safety and hazard analysis [LEV95] are software quality assurance activities (Chapter 8) that focus on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail. If hazards can be identified early in the software engineering process, software design features can be specified that will either eliminate or control potential hazards.

6.8 THE RMMM PLAN

A risk management strategy can be included in the software project plan or the risk management steps can be organized into a separate Risk Mitigation, Monitoring and Management Plan. The RMMM plan documents all work performed as part of risk analysis and are used by the project manager as part of the overall project plan.

Some software teams do not develop a formal RMMM document. Rather, each risk is documented individually using a risk information sheet (RIS) [WIL97]. In most cases, the RIS is maintained using a database system, so that creation and information entry, priority ordering, searches, and other analysis may be accomplished easily. The format of the RIS is illustrated in Figure 6.5.

Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence. As we have already discussed, risk mitigation is a problem avoidance activity. Risk monitoring is a project tracking activity with three primary objectives: (I) to assess whether predicted risks do, in fact, occur; (2) to ensure that risk aversion steps defined for the risk are being properly applied; and (3) to collect information that can be used for future risk analysis. In many cases, the problems that occur during a project can be traced to more than one risk. Another job of risk monitoring is to attempt to allocate origin (what risk(s) caused which problems throughout. the project).

Risk management is not as well developed as are some of the more traditional project management disciplines. Some organizations feel that risk management is too specialized or
advanced for them. Others believe that risk management is optional. Others fear that risk management may expose flaws in their project plans and strategies that will hurt them in the competitive world. Certainly these organizations will not talk openly and honestly about risk and will “shoot the messenger” that brings risk to their attention.
Risk-mature organizations overcome the barriers to practicing effective risk management in their approach to projects.

- Their organizational culture becomes “risk friendly.”
- Risk management gains priority to rank along with cost, time and scope management.
- Decisions are made and resources are allocated based on the results of risk analysis.
- The highest quality data are used for risk analysis and resources are committed to the efforts.
- Risk management is viewed as a career path in the organization and those that practice it are treated as professionals.
- Risk analysis functions are given independence in the organization even though that may make it hard to “control.”
- Mature risk management organizations look to the best in class to benchmark their risk management processes.
- They use modern tools and are not disdainful of sophisticated and proven approaches.
- They measure their effectiveness with metrics.
- Project decisions are made on a “risk-adjusted” basis.
- Continuous improvement is achieved through regular repetition.

They participate in professional interchanges through conferences and journals, sharing what they have learned.

Project risk is an uncertain event or condition that, if it occurs, has a positive or a negative effect on at least one project objective. A risk may have one or more causes and, if it occurs, one or more impacts. For example, a cause may be requiring an environmental permit to do work or having limited personnel assigned to design the project. The risk event is that the permitting agency may take longer than planned to issue a permit for some reason, or the design personnel available and assigned may not be adequate for the task. If either of these uncertain events occurs, there may be an impact on the project cost, schedule, or performance. Risk conditions could include aspects of the project's or organization's environment that may contribute to project risk, such as poor project management practices, lack of integrated management systems, concurrent multiple projects, or dependency on external participants who cannot be controlled.

Project risk has its origins in the uncertainty that is present in all projects. Known risks are those that have been identified and analyzed, and it may be possible to plan for those risks using the processes described in this chapter. Unknown risks cannot be managed proactively, and a prudent response by the project team can be to allocate general contingency against such risks, as well as against any known risks for which it may not be cost-effective or possible to develop a proactive response.

Organizations perceive risk as it relates to threats to project success or to opportunities to enhance chances of project success. Risks that are threats to the project may be accepted if the risk is in balance with the reward that may be gained by taking the risk. For example, adopting a fast track schedule (Section 6.4) that may be overrun is a risk taken to achieve an earlier completion date. Risks that are opportunities, such as work acceleration that may be gained by assigning more expert staff, may be pursued to benefit the project's objectives.

Individuals, and by extension organizations, have attitudes toward risk that affect both the accuracy of the perception of risk and the way they respond. Attitudes about risk should be made explicit wherever possible. A consistent approach to risk that meets the organization's requirements should be developed for each project, and communication about risk and its handling should be open and honest. Risk responses reflect an organization's balance between risk-taking and risk-avoidance.

To be successful, the organization must be committed to addressing the management of risk proactively and consistently throughout the project.

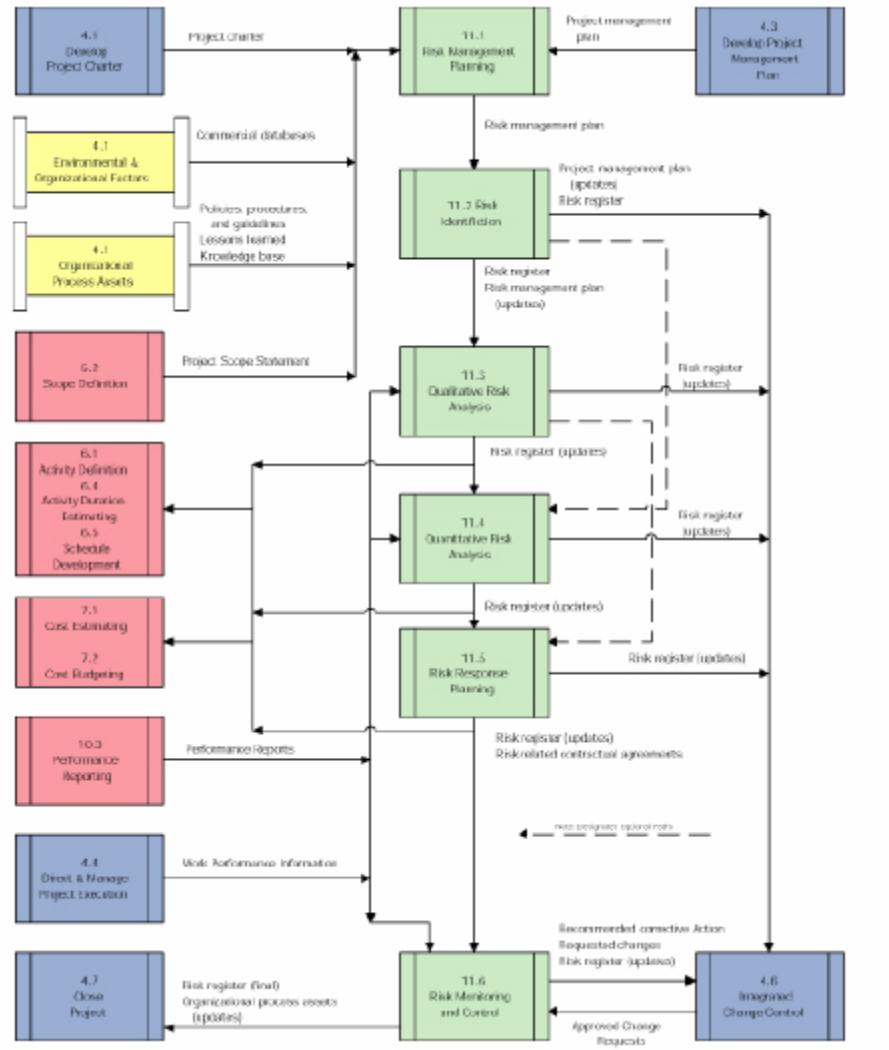


Figure: Project Risk Management Flow

LECTURE # 40

9. Risk and Change Management

9.1 Fundamentals

- **What is it?**

  Change Management is the process by which changes to the Project's scope, deliverables, timescales or resources are formally defined, evaluated and approved prior to implementation.

  A core aspect of the Project Manager's role is to manage change within the project successfully. This is achieved by understanding the business and system drivers requiring the change, documenting the benefits and costs of adopting the change and formulating a structured plan for implementing the change.

  To formally request a change, it is often necessary to complete a Change Form. The change request details may then be recorded within a Change Register.

  Risk Management is the process by which risks to the project (e.g. to the scope, deliverables, timescales or resources) are formally identified, quantified and managed during the project.

  A project risk may be identified at any stage of the project by completing a Risk Form and recording the relevant risk details within the Risk Register.

  First, risk concerns future happenings. Today and yesterday are beyond active concern, as we are already reaping what was previously sowed by our past actions. The question is; can we, therefore, by changing our actions today, create an opportunity for a different and hopefully better situation for ourselves tomorrow.

  This means, *second*, that risk involves change, such as in changes of mind, opinion, actions, or places…

  [*Third*] risk involves choice, and the uncertainty that choice itself entails. Thus paradoxically, risk, like death and taxes, is one of the few certainties of life.

- **Who does it?**
  Everyone involved in the software process – managers, software engineers and customers - participate in risk analysis and management

- **Why is it important?**

Think about the Boy Scout motto: "Be prepared" Software is a difficult undertaking. Lots of things can go wrong, and frankly, many often do. It's for this reason that being prepared, understanding the risks and taking preventive measures to avoid or manage them is a key element of good software project management.

- **What are the steps?**
  Recognizing what can go wrong is the first step, called - risk identification. Next, each risk is analyzed to determine the likelihood that it will occur and the damage that it will do if it does occur. Once this information is established, risks are ranked by probability and impact. Finally, a plan is developed to manage those risks with high probability and high impact.

- **What is the work product?**
  A risk mitigation monitoring and management (RMMM) plan or a set of risk information sheets is produced.

- **How do I ensure that I've done it right?**
  The risks that are analyzed and managed should be derived from thorough study of the people, the product, the process and the project. The RMMM should be revisited as the project proceeds to ensure that risks are kept up to date. Contingency plans for risk management should be realistic.

9.2 Risk & Change Management Concepts

Foresight is an excellent management quality that can often be cultivated with experience. Indeed, in many cases, problems can be anticipated.

In such cases, the manager can plan for the possibility that a problem will occur by estimating its probability, evaluating its impact, and preparing solutions in advance. This is referred to as an effective means of combating potential development problems.

Performing risk analysis means being prepared. It is a form of insurance, the basic idea being that if a problem occurs, a solution is readily available. Like all insurance, risk analysis usually comes with a price.

The cost of preparing for the occurrence of a problem is primarily the cost of having the alternative solution at hand while the problem may or may not occur.

In some cases, the cost may be minimal, the time needed to analyze and document the solution, and the time to track the problem.

In other cases the cost may be substantial, for example, the price of an alternative piece of development equipment.

In any case, a problem that has been analyzed and resolved ahead of time is far simpler to resolve than a problem that occurs unexpectedly.

When risk is considered in the context of software engineering, three conceptual underpinnings are always in evidence:

- The future is our concern – what risks might cause the software project to go awry?
- Change is our concern -how will changes in customer requirements, development technologies, target computers, and all other entities connected to the project affect timeliness and overall success?
- Last, we must grapple with choices - what methods and tools should we use, how many people should be involved, how much emphasis on quality is "enough"?

Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty. Many problems can plague a software project

A risk is a potential problem - it might happen, it might not But regardless of the outcome, it's a really good idea to identify it, assess its probability of occurrence, estimate its impact, and establish a contingency plan should the problem actually occur.

Any project can encounter uncertainties in the form of increased costs, schedule delays, and diminished qualities. Unless tackled, these uncertainties can lead to major project disasters.

The uncertainties encountered during project execution are the potential project risks. Every software project has to grapple with the new risks threatening information security along with the conventional risks, such as hardware failure, time and cost escalation, defects, or resource crunch.

Risk can be defined as the possibility of loss. Risk arises due to the inability to achieve objectives within defined cost, schedule, and technical constraints.

Risk has two components:

- The possibility of not achieving a particular outcome is one, and
- The result of failing to achieve the outcome is the other

The former is the probability of loss, and the latter is the loss. Software project management deals with managing both these components of risk.

Risk management focuses the project manager’s attention on those portions of the project most likely to cause trouble and compromise participants’ win conditions.

In other words, risk management is a set of actions that helps the project manager plan to deal with uncertain occurrences. It is through risk management that project managers assess risks and manage to reduce risks to an acceptable level.

9.3 Types of Risks

To be able to manage project risks, you must first understand what constitutes, a risk. All uncertain occurrences are not risks.

Only those occurrences that have an adverse impact on the progress of a project are risks to the project.

Risk is not a bad thing. Risk is bad only when it results in loss for an organization. Unless there is a potential for loss, there is no risk.

Moreover, loss can be interpreted as either a bad outcome or a lost opportunity. The tendency of most project managers is to jump at the statement this is a risk.

However, the desired reaction is to pre-empt all possible outcome and plan for them. Project risks can be broadly categorized into development process risks and product risks.

i. Development Process Risks

The risks encountered during product development are categorized as development process risks.

These comprise developer errors, natural disasters, disgruntled employees, and poor management objectives.

Developer errors could be attributed to poor training due to budgetary constraints and inadequate skills and software tools.

Ergonomic problems, environment problems, and interruptions or distractions at office also account for developer risks.

Other risks in this category include problems in personnel acquisition and retention.

Similarly, natural disasters such as flood, cyclone, fire, storm, and snowfall are also risks to a project.

Disgruntled employees can also become a risk to an organization. For example, a sacked employee can use password snuffers to gain unauthorized access. A dismissed person can flood the system with senseless messages. A

disgruntled employee can also try to sabotage the project work by destroying files and programs.

A poorly defined management objective is another development process risk. If the language in the management objective is ambiguous and not stated clearly, the risk management program will not function properly.

Narrowly focused and changing objectives that are not updated can also be counted as risks.

Lack of contingency plans, incomplete cost estimates, and unrealistic schedules are also potential risks in a project.

Similarly, unrealistic performance standards are also potential risks to the development process.

Other possible risks include contractual risks, technological risks, and inadequate documentation of other concurrent projects.

ii. Product Risks

Product risks crop up in the form of changing requirements during product development.

Incomplete and unclear requirements are a risk to the product during development.

Similarly, problems in meeting design specifications can also be categorized as risk to product development.

Risks could arise if the project deliverables or objectives are not clearly defined or if technical data is missing.

The possibility of several alternatives at any given time during the project is also a cause of concern. If errors are not recognized during the design phase, they could turn into risks for the project.

Similarly, risks could arise due to the size and complexity of the product or while achieving client acceptance of the product.

Note:

The key idea in risk management is not to wait for a risk to materialize and become a problem. The objective of risk management is to ensure that for each perceived risk; you know well in advance how to tackle it.

9.4 Risk Management Process

The process of risk management begins during the analysis phase of software development life cycle. However, the actual process of managing risks continues throughout the product development phase.

Risk management is a dynamic process because it deals with the activities that are yet to happen.

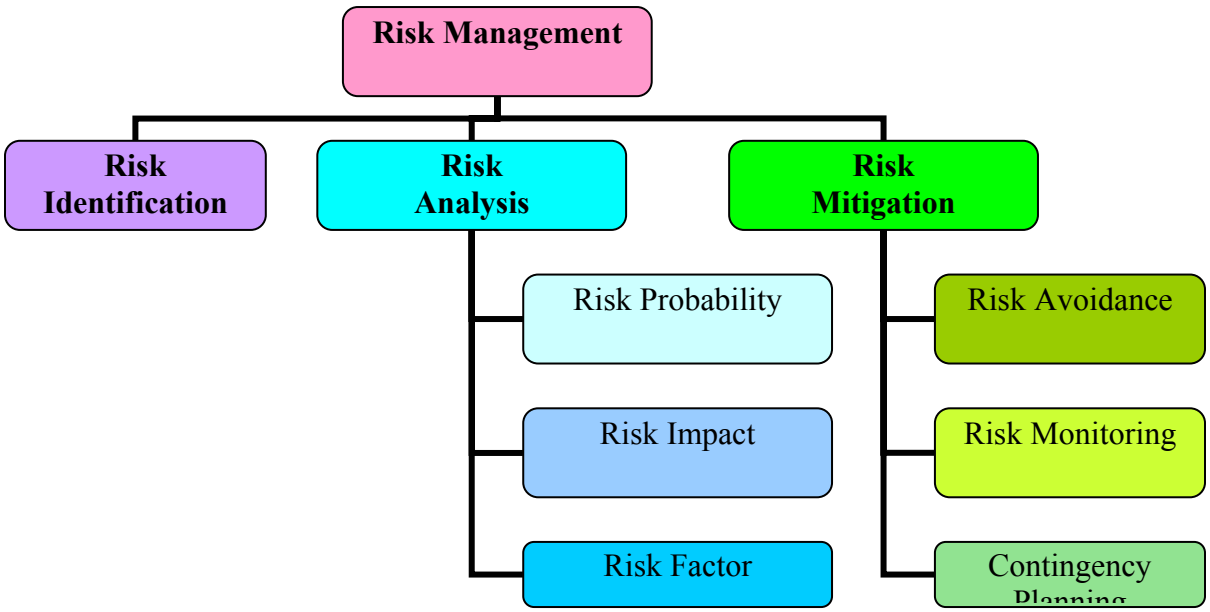
Risk management has a two-fold agenda. First, deciding actions for preventing risks from happening, and second, deciding actions for tackling risks that materialize.

Therefore, risk management is all about pre-empting a risk, coming up with a plan for resolving the risk, and finally executing the plan.

Figure 1 displays the steps of the risk management process. Formally, articulated, risk management process consists of three steps:

4. *Risk identification*
5. *Risk analysis*
6. *Risk mitigation*

Figure 1: Risk Management Process



LECTURE# 41

9. Risk and Change Management

9.4 Risk Management Process

i. Risk Identification

Risk Identification involves:

– Identifying risks that may occur on a particular project
– Determining which risks might affect the project
– Documenting their characteristics

Participants in risk identification activities can include the following, where appropriate:

- Project manager
- Project team leaders
- Project team
- Risk management team if assigned
- Subject matter experts from outside the project team
- Customers
- End users
- Other project managers
- Stakeholders
- Outside risk management experts

Risk Identification is an iterative process because new risks may become known as the project progresses through its life cycle. The frequency of iteration and who participates in each cycle will vary from case to case.

The project team should be involved in the process so that they can develop and maintain a sense of ownership of and responsibility for the risks and associated risk response actions. Persons outside the team may provide additional objective information.

The Risk Identification process usually leads to the Qualitative Risk Analysis process. Alternatively, it can lead directly to the Quantitative Risk Analysis process when conducted by an experienced risk manager. On some occasions simply the identification of a risk may suggest its response, and these should be recorded for further analysis and implementation in the Risk Response Planning process.

In this step, the project manager gathers information about the potential risks in the project. The project manager plans the strategies for avoiding risks or controlling them.

The project team conducts brainstorming sessions and discussions among team members about the requirements document. They discuss the available technology, manpower, prevailing environment, and all project-related factors. The project manager picks up the thread from these and creates a risk log.

After the risk log is prepared, the project manager calls a meeting within the team and technical experts to discuss the risk log and the mitigation plans. An effective way of identifying' risks is using a questionnaire.

One method for identifying risks is to create a risk item checklist. The checklist can be used for risk identification and focuses on some subset of known and predictable risks in the following generic subcategories:

a) **Product size-risks** associated with the overall size of the software to be built or modified.

b) **Business impact-risks** associated with constraints imposed by management or the marketplace.

c) **Customer characteristics-risks** associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.

d) **Process definition-risks** associated with the degree to which the software process has been defined and is followed by the development organization.

e) **Development environment-risks** associated with the availability and quality of the tools to be used to build the product.

f) **Technology to be built-risks** associated with the complexity of the system to be built and the newness of the technology that is packaged by the system.

g) **Staff size and experience-risks** associated with the overall technical and project experience of the software engineers who will do the work.

The risk item checklist can be organized in different ways. Questions relevant to each of the topics can be answered for each software project.

The answers to these questions allow the planner to estimate the impact of risk. A different risk item checklist format simply lists characteristics that are relevant to each generic subcategory.

Finally, a set of "risk components and drivers" are listed along with their probability of occurrence.

A number of comprehensive checklists for software project risk have been proposed in the literature (e.g., [SEI93], [KAR96]). These provide useful insight into generic risks for software projects and should be used whenever risk analysis and management is instituted. However, a relatively short list of questions [KEI98] can be used to provide a preliminary indication of whether a project is "at risk."

During risk identification, you obtain answers to the following queries:

a. *Why is the risk important?*
b. *What information is needed to track the status of the risk?*
c. *Who is responsible for the risk management activity?*
d. *What resources are needed to perform the activity?*
e. *What is the detailed plan to improve the risk and / or mitigate it?*

Table 1 displays a sample risk identification questionnaire.

Table 1: Sample Risk Identification Questionnaire

SN		Yes/No/Not Applicable (NA)
A.	**Risk Description**	
1)	Are requirements changing continuously during product development?	
2)	Do the changing requirements affect each of the following:	
	Quality	
	Functionality	
	Schedule	
	Integration	
	Design	
	Testing	
3)	Are the external interfaces changing?	
4)	Are the requirements missing or incompletely specified?	
5)	Are there any missing requirements?	
6)	Can these requirements be incorporated into the system?	
7)	Does the client have unwritten requirements or expectations?	

8)	Is there a way to capture these requirements?	
9)	Are requirements unclear or in need of interpretation?	
10)	Are you able to understand the requirements as written?	
11)	Will the requirements lead to the product the client wants?	
12)	Are there any requirements that may not specify what the client really wants?	
B.	**Product Design**	
1)	Do you encounter problem in meeting functionality requirements?	
2)	Are there any specified algorithms that may not satisfy the requirements?	
3)	Will the design and/or implementation be difficult to achieve?	
4)	Are there any requirements or functions that are difficult to design?	
C.	**Reusability**	
1)	Are you reusing the program?	
2)	Do you foresee any problems in documentations?	
3)	Do you foresee any problems in performance?	
4)	Do you foresee any problems in functionality?	

The sample questionnaire in Table 1 includes an exhaustive list of risks that might be encountered during the progress of a project.

The answers to the questions in the risk identification questionnaire enable the project manager to estimate the impact of risks.

In the sample questionnaire Table 1, the risks are categorized under product engineering, product-design, and reusability.

From this table, you obtain a list of risks that are relevant to each category. You compare the information obtained from this table with past results and estimate the criticality of risk.

Risk identification is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.).

By identifying known and predictable risks, the project manager takes a first step toward avoiding them when possible and controlling them when necessary.

There are two distinct types of risks for each of the categories that have been presented in Section 6.2: generic risks and product-specific risks. Generic risks are a potential threat to every software project. Product-specific risks can be

identified only by those with a clear understanding of the technology, the people, and the environment that is specific to the project at hand. To identify product - specific risks, the project plan and the software statement of scope are examined and an answer to the following question is developed: What special characteristics of this product may threaten our project plan?

Risk Identification involves identifying risks that may occur on a particular project and determining which risks might affect the project and documenting their characteristics.

ii. Risk Analysis

After identifying the risks, the project manager needs to analyze the risks. Uncertainty and loss are the two characteristics of risk.

The uncertainty factor in risk means that the unknown event mayor may not happen. While analyzing risks, the project manager needs to quantify the level of uncertainty and the degree of loss.

Based on this, the project manager plans schedules and costs. During analysis, information on risk is converted into information on decision-making. Analysis provides the basis for the project manager to work on the "right" risks.

Table 2: Risk Analysis Table

Risk Description	Probability of Occurrence (0 – 1)	Impact on Project (1 – 10)	Risk Factor (Probability x Impact)

Table 2 displays a risk analysis format. There are various tasks involved in risk analysis.

First, the WBS elements are identified. One of the tasks in the risk analysis phase is to describe the risk. The risk can be product-related, process-related, organization-related, client-related, or infrastructure-related.

Second, the WBS elements are evaluated to determine the risk events.

Then the project manager quantifies the probability of occurrence of risk. The project manager can assign probability values between 0 and 1. For example, a risk with a low probability of occurrence is marked 0.2 while that with a high

probability of occurrence is marked 0.8. The reason why a particular risk has a high or low probability depends on the actual circumstance of the project.

Third, the risks are rated depending on their probability of occurrence. Based on the probability of risk, the project manager identifies the impact of the risk. The impact of risk on cost, schedule, and quantity needs to be calculated and graded. The impact of risk can be graded on a scale of 1 to 10, 1 being the lowest, and 10 being the highest.
Then the risk factor is calculated by multiplying the probability of risk and the impact of risk. Finally, each risk is prioritized relative to other risks. The risk factor is used to prioritize the identified risks. For example, the risk with a probability value 0.1 and an impact value 2 will have minimal impact. While risks close to probability value 0.8 and with an impact value 9 will have greater impact. Therefore, the project manager can prioritize risks based on the probability and the impact of risks. A risk that has a high impact and low probability will not absorb a significant amount of the project manager's time. However, high-impact risks with moderate to high probability will catch the attention of the project manager.

iii. Risk Mitigation

Risk mitigation is the best possible approach adopted by the project manager to avoid risks from occurring. The probability of the risk occurring and the potential impact of the risk can be mitigated by dealing with the problem early in the project.
Essentially, risk mitigation involves three possibilities and the project manager needs to adopt a risk mitigation strategy aimed at them. The three possibilities include:

1. Risk avoidance
2. Risk monitoring
3. Contingency planning

1. Risk Avoidance

To avoid risks from occurring, the project team prepares the risk plan before the commencement of the project. The project team identifies the potential risks and prioritizes them based on their probability of occurrence and impact. Then, the team prepares a plan for managing risks. In most software projects, this plan is popularly called the risk management plan. Table 3 displays the format of a risk management plan.

Table 3: Risk Management Plan

Risk Description	Probability of Occurrence (0 – 1)	Impact on Project (1 – 10)	Risk Factor (Probability x Impact)	Mitigation Steps	Responsibility	Start Date	End Date

To prepare the risk management plan, the project team first identifies and assesses the risks associated with the project.

Then, the probability of occurrence of each risk is estimated and the possible impact is calculated.

In the plan, the probable cost and damage is also quantified.

The project team also identifies the contingency plans for all the identified risks.

The contingency plan for each risk is based on the project's defined operational software process. The plan is modified throughout the software development life cycle of the project based on the changes taking place.

The contingency plan also includes the cost, in terms of effort, in carrying out the plan. The software risks identified are tracked, reassessed, and re-planned at the end of each phase.

The project manager revisits the plan if significant changes are introduced in the software project.

2. Risk Monitoring

As the project proceeds, risk-monitoring activities commence. It is not possible to monitor closely all the risks that are identified for the project. For example, if 100 risks are identified for a project, only top 20 risks are monitored.

There are re-planning checkpoints where the information obtained from monitoring the risks is used to refine the risk assessments and management plan.

The project manager monitors the top 20 percent of the factors that may indicate the status of the risks in the project.

In the case of large teams, the project manager also needs to monitor the attitude of the team members and their problems. This helps the project manager monitor any possible team-related risks.

Besides monitoring the top 20 percent of the risks, the project manager needs to monitor the mitigation steps also.

Consider an example. To ensure that particular software is browser- independent, the software is created on the lowest compatible browser. Such software will work on any browser thus making it browser-independent.

Therefore, mitigating a project risk involves working hard at reducing the possibility that the risk will ever occur.

Mitigation includes nearly all actions that a project team takes to overcome risks. For example, choosing a more expensive but proven technology over, a newer, less expensive technology is a step toward mitigating project risks.

3. Contingency Planning

The possibility of contingency planning arises when mitigation efforts fail and risk becomes a reality.

Contingency planning is used to monitor risks and invoke a predetermined response. According to the plan, a trigger is set up. If the trigger is reached, the contingency plan is put into effect.

Contingency planning involves maintaining an alternative plan if the original plan fails. A simple example could be the savings people make for a rainy day. Contingency plans are a must for the top 20 percent of the risks identified. These plans are put to use after the risks become a reality.

The importance of contingency planning can be realized from this example. Despite the massive attack on WTC, the stock markets in the US resumed functioning within a few days. This was possible because the finance companies had backed up their data and information on computers elsewhere. The contingency planning of finance companies prevented the risk of huge data loss for the stock market.

iv. Managing Risks: An Example

Consider a scenario. Your organization is a vendor of software solutions. A bus transport company the US wants you to develop a Schedule Adherence system. The team that will develop this software is new and the platform selected for development is also new to your organization. The project team needs to be trained intensively for this.

During this project, the team is expected to manage a large volume of data. The team has never had any experience in managing such a large volume of data. The system also needs to use this data to generate various MIS reports related to delays or adherence of bus services.

The performance requirement is less than fifteen seconds for all popular browsers. Your organization is anticipating numerous requirement changes during the

development process. The system needs to be implemented across several states in the country. The data related to the system is highly confidential because it can provide an edge to the competitors.

Now, as a project manager, you need to prepare a risk management plan for this project. The project starts on May 15 and should be completed on November 15.

First, you need to identify the potential risks involved in the project. The potential risks to the project are described in Table 4.

Table 4: Potential Risks to a Project

Risk Description
Inexperienced staff
Performance risk due to high volume of data to be processed
Cross-browser compatibility
Involvement of new technology
Design changes during development

After identifying the risks, you need to estimate the probability of their occurrence and their impact on product development. Based on this, you calculate the risk factor and plan the mitigation steps. Your risk management plan is displayed in Table 5.

Table 5: The Risk Management Plan for Building a Schedule Adherence System

Risk Description	**Probability of Occurrence (0 – 1)**	**Impact on Project (1 – 10)**	**Risk Factor (Probability x Impact)**	**Mitigation Steps**	**Responsibility**	**Start Date**	**End Date**
Inexperience Staff	0.8	3	2.4	Project Manager	Conducting training sessions before the need commencement of project	May 15	July 15
Performance risk due to high volume of data to be processed	0.6	7	4.2	Architect	Massive tuning of architecture during the design phase and conducting	May 15	Till the end of the project

					a proof of concepts for the design		
Cross-browser compatibility	0.6	5	3.0	Developer	Using the lowest compatible browser for development	May 15	Till the end of the project
Involvement of new technology	0.5	5	2.5	Project Manager	Ensuring all details pertaining to the technology is available and keeping in close touch with the technology vendor	May	Till the end of the project
Design changes during development	0.6	8	4.8	Architect	Designing a flexible architecture that can accommodate future changes and enhancement	May 15	Till the end of the project

In Table 5, the risk factors with high values are the high priority risks. Here, the high priority risks are design changes during development, performance risk due to high volume of data to be processed, and cross-browser compatibility.

The high priority risks need to be monitored closely and continuously. The rest of the risks can be monitored periodically.

Certain risks need contingency planning despite being low in the priority list. For example, the risk due to involvement of new technology is a low priority risk.

However, the probability of occurrence of this risk is 0.5, which is fairly high. This calls for contingency planning because you may not be aware of all the details of the new technology. Suppose the system fails to respond if the number of users exceeds a certain number. In such a situation, you need to have a contingency plan ready. You need to discuss with the vendor regarding a workaround for such a situation. The workaround should be ready in the case of failure of the system.

9.5 Risk Components and Drivers

The U.S. Air Force [AFC88j has written a pamphlet that contains excellent guidelines (or software risk 1dentification and abatement. The Air Force approach requires that the project manager identify the risk drivers that affect software risk components- Performance, cost, support, and schedule. In the context of this discussion, the risk components are defined in the following manner:

a) **Performance risk** - the degree of uncertainty that the product will meet its requirements and be fit for its intended use.
b) **Cost risk** - the degree of uncertainty that the project budget will be maintained.
c) **Support risk** - the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
d) **Schedule risk** - the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

The impact of each risk driver on the risk component is divided into one of four impacts categories-negligible, marginal, critical, or catastrophic.

Referring to Figure 2, a characterization of the potential consequences of errors (rows labeled 1); or a failure to achieve a desired outcome (rows labeled 2) are described. The impact category is chosen based on the characterization that best fits the description in the table.

Figure 2: Impact assessment

Components/Category		**Performance**	**Support**	**Cost**	**Schedule**
Catastrophic	1	Failure to meet the requirement would result in mission failure		Failure results in increased costs and schedule delays with expected values of $500 K	
	2	Significant degradation to non achievement of technical performance	Non responsive or unsupportable software	Significant financial shortages, budget overrun likely	Unavoidable IOC
Critical	1	Failure to meet the requirement would degrade system performance to a point where mission success is questionable		Failure results in operational delays and/or increased costs with expected $100 K to $ 500 K	

	2	Some reduction in technical performance	Minor delays in software modification	Some shortage of financial resources, possible overruns	Possible slippage in IOC
Marginal	1	Failure to meet the requirement would result in degradation of secondary mission		Costs, impacts and / or recoverable schedule slips with expected value of $ 1 K to $ 100 K	
	2	Minimal to small reduction in technical performance	Responsive software support	Sufficient financial resources	Realistic achievable schedule
Negligible	1	Failure to meet the requirement would create inconvenience or non operational impact		Error results in minor cost and / or schedule impact with expected value of less than $ 1 K	
	2	No reduction in technical performance	Easily supportable software	Possible budget under run	Early achievable IOC

Note:

(3) The potential consequence of undetected software errors or faults.

(4) The potential consequence if the desired outcome is not achieved.

9.6 Developing a Risk Table

A risk table provides a project manager with a simple technique for risk projection. A sample risk table is illustrated below.

Sample risk table prior to sorting

Risks	Category	Probability	Impact	RMMM
Size estimate may be significantly low	PS	60%	2	
Larger number of users than planned	PS	30%	3	
Less reuse than planned	PS	70%	2	
End-users resist system	BU	40%	3	
Delivery deadline will be lightened	BU	50%	2	
Funding will be lost	CU	40%	1	
Customer will change requirements	PS	80%	2	
Technology will not meet expectations	TE	30%	1	
Lack of training on tools	DE	80%	3	
Staff inexperienced	ST	80%	2	
Staff turnover will be high	ST	60%	2	

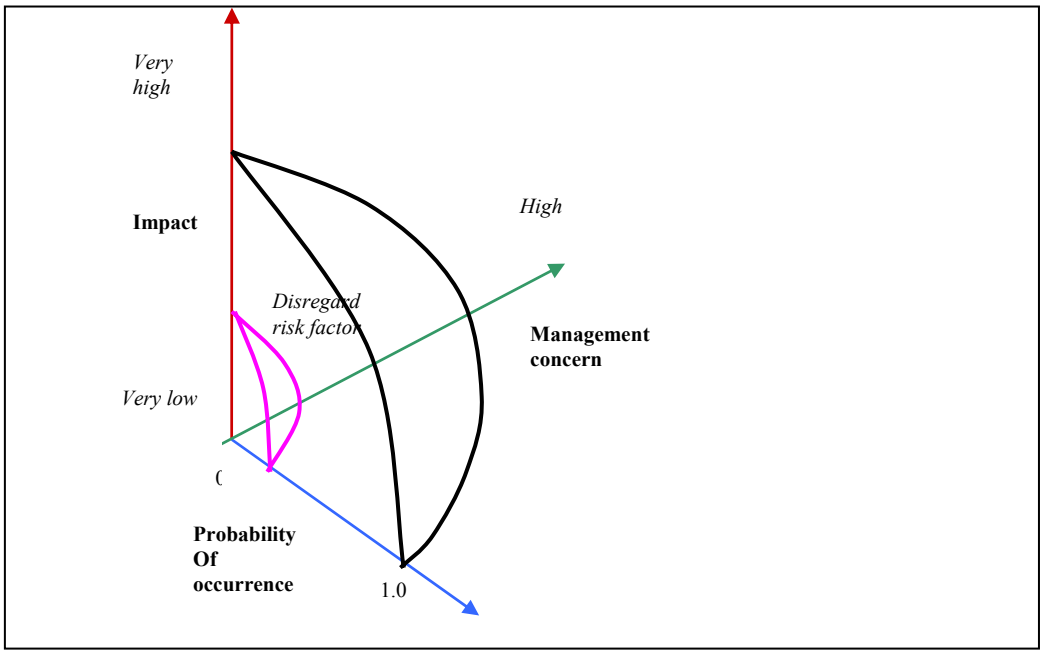
Impact values:
l- Catastrophic
2- Critical
3- marginal
4- Negligible

A project team begins by listing all risks (no matter how remote) in the first column of the table. This can be accomplished with the help of the risk item check-lists given earlier.

Each risk is categorized in the second column (e.g., PS implies a project size risk, BU implies a business risk). The probability of occurrence of each risk is entered in the next column of the table. The probability value for each risk can be estimated by team members individually. Individual team members are polled in round-robin fashion until their assessment of risk probability begins to converge.

Next, the impact of each risk is assessed. Each risk component is assessed using the characterization presented in the sample risk table, and an impact category is determined. The categories for each of the four risk components - *performance, support, cost, and schedule* - are averaged to determine an overall impact value.

Figure 2: Risk and management concern



Once the first four columns of the risk table have been completed, the table is sorted by probability and by impact. High-probability, high-impact risks percolate to the top of the table, and low-probability risks drop to the bottom. This accomplishes first order risk prioritization. The project manager studies the

resultant sorted table and defines a cutoff line. The cutoff line (drawn horizontally at some point in the table) implies that only risks that lie above the line will be given further attention. Risks that fall below the line are re-evaluated to accomplish second-order prioritization.

Referring to Figure 2, risk impact and probability have a distinct influence on management concern. A risk factor that has a high impact but a very low probability of occurrence should not absorb a significant amount of management time. However, high-impact risks with moderate to high probability and low-impact risks with high probability should be carried forward into the risk analysis steps that follow.

All risks that lie above the cutoff line must be managed. The column labeled RMMM contains a pointer into Risk Mitigation, Monitoring and Management Plan or alternatively, a collection of risk information sheets developed for all risks that lie above the cutoff.

9.7 Reactive VS. Proactive Risk Strategies

Reactive strategies have been laughingly called the "Indiana Jones School of risk management" [THO92]. In the movies that carried his name, Indiana Jones, when faced with overwhelming difficulty, would invariably say, "Don't worry, I'll think of something!" Never worrying about problems until they happened, Indy would react in some heroic way.

Sadly, the average software project manager is not Indiana Jones and the members of the software project team are not his trusty sidekicks. Yet, the majority of software teams rely solely on reactive risk strategies. At best, a reactive strategy monitors the project for likely risks. Resources are set aside to deal with them, should they become actual problems. More commonly, the software team does nothing about risks until something goes wrong. Then, the team flies into action in an attempt to correct the problem rapidly. This is often called a fire fighting mode. When this fails, "Crisis Management" [CHA92] takes over, and the project is in real jeopardy.

A considerably more intelligent strategy for risk management is to be proactive. A proactive strategy begins long before technical work is initiated. Potential risks are identified, their probability and impact are assessed and they are ranked by importance. Then, the software team establishes a plan for managing risk. The primary objective is to avoid risk, but because not all risks can be avoided, the team works to develop a contingency plan that will enable it to respond in a controlled and effective manner.

9.8 Risk Mitigation, Monitoring, and Management

All of the risk analysis activities presented to this point have a single goal to assist the project team in developing a strategy for dealing with risk. An effective strategy must consider three issues:

- risk avoidance
- risk monitoring
- risk management and contingency planning

If a software team adopts a proactive approach to risk, avoidance is always the best strategy. This is achieved by developing a plan for risk mitigation. For example, assume that high staff turnover is noted as a project risk, r_l Based on past history and management intuition, the likelihood, l_l of high turnover is estimated to be 0.70 (70 per cent, rather high) and the impact, x_l is projected at level 2. That is, high turnover will have a critical impact on project cost and schedule.

To mitigate this risk, project management must develop a strategy for reducing turnover. Among the possible steps to be taken are

- Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, and competitive job market).
- Mitigate those causes that are under our control before the project starts.
- Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.
- Organize project teams so that information about each development activity is widely dispersed.
- Define documentation standards and establish mechanisms to be sure that documents are developed in a timely manner.
- Conduct peer reviews of all work (so that more than one person is “up to speed").
- Assign a backup staff member for every critical technologist.

As the project proceeds, risk monitoring activities commence. The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely. In the case of high staff turnover, the following factors can be monitored:

- General attitude of team members based on project pressures.
- The degree to which the team has jelled.
- Interpersonal relationships among team members.
- Potential problems with compensation and benefits.
- The availability of jobs within the company and outside it.

In addition to monitoring these factors, the project manager should monitor the effectiveness of risk mitigation steps. This is one mechanism for ensuring continuity, should a critical individual leave the project. The project manager

should monitor documents carefully to ensure that each can stand on its own and that each imparts information that would be necessary if a newcomer were forced to join the software team somewhere in the middle of the project.

Risk management and contingency planning assumes that mitigation efforts have failed and that the risk has become a reality. Continuing the example, the project is well underway and a number of people announce that they will be leaving. If the mitigation strategy has been followed, backup is available, information is documented, and knowledge has been dispersed across the team. In addition, the project manager may temporarily refocus resources (and readjust the project schedule) to those functions that are fully staffed, enabling newcomers who must be added to the team to "get up to speed." Those individuals who are leaving are asked to stop all work and spend their last weeks in "knowledge transfer mode. This might include video-based knowledge capture, the development of "commentary documents," and/or meeting with other team members who will remain on the project.

It is important to note that RMMM steps incur additional project cost. For example, spending the time to "backup" every critical technologist costs money. Part of risk management, therefore, is to evaluate when the benefits accrued by the RMMM steps are outweighed by the costs associated with implementing them. In essence, the project planner performs a classic cost/benefit analysis. If risk aversion steps for high turnover, it will increase both project cost and duration by an estimated 15 percent, but the predominant cost factor is "backup," management may decide not to implement this step. On the other hand, if the risk aversion steps are projected to' increase costs by 5 percent and duration by only 3 percent management will likely put all into place.

For a large project, 30 or 40 risks may be identified. If between three and seven risk management steps are identified for each, risk management may become a project in itself! For this reason, we adapt the Pareto 80-20 rule to software risk. Experience indicates that 80 percent of the overall project risk (i.e., 80 percent of the potential for project failure) can be accounted for by only 20 percent of the identified risks. The work performed during earlier risk analysis steps will help the planner to determine which of the risks reside in that 20 percent (e.g., risks that lead to the highest risk exposure. For this reason, some of the risks identified, assessed, and projected may not make it into the RMMM plan - they don't fall into the critical 20 percent (the risks with highest project priority).

9.9 SAFETY RISKS AND HAZARDS

Risk is not limited to the software project itself. Risks can occur after the software has been successfully developed and delivered to the customer. These risks are typically associated with the consequences of software failure in the field.

In the early days of computing, there was reluctance to use computers (and software) to control safety critical processes such as nuclear reactors, aircraft flight control, weapons systems, and large-scale industrial processes. Although the probability .of failure of a well-engineered system was small, an undetected fault in a computer- based control or monitoring system could result in enormous economic damage or, worse, significant human injury or loss of life. But the cost and functional benefits of Computer-based control and monitoring far outweigh the risk. Today, computer hardware and software are used regularly to control safety critical systems.

When software is used as part of a control system, complexity can increase by an order of magnitude or more. Subtle design faults induced by human error-something that can be uncovered and eliminated in hardware-based conventional control-become much more difficult to uncover when software is used.

Software safety and hazard analysis [LEV95] are software quality assurance activities that focus on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail. If hazards can be identified early in the software engineering process, software design features can be specified that will either eliminate or control potential hazards.

9.10 THE RMMM PLAN

A risk management strategy can be included in the software project plan or the risk management steps can be organized into a separate Risk Mitigation, Monitoring and Management Plan. The RMMM plan documents all work performed as part of risk analysis and are used by the project manager as part of the overall project plan.

Some software teams do not develop a formal RMMM document. Rather, each risk is documented individually using a risk information sheet (RIS) [WIL97]. In most cases, the RIS is maintained using a database system, so that creation and information entry, priority ordering, searches, and other analysis may be accomplished easily. The format of the RIS is illustrated in Figure 6.5.

Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence. As we have already discussed, risk mitigation is a problem avoidance activity. Risk monitoring is a project tracking activity with three primary objectives:

1. To assess whether predicted risks do, in fact, occur.
2. To ensure that risk aversion steps defined for the risk are being properly applied.
3. To collect information that can be used for future risk analysis.

In many cases, the problems that occur during a project can be traced to more than one risk. Another job of risk monitoring is to attempt to allocate origin (what risk(s) caused which problems throughout. the project).

LECTURE # 42

10.Quality

10.1 Quality Concept

What is it? It's not enough to talk the talk by saying that *soft* ware quality is important, you have to (1) explicitly define what is meant when you say 'software quality, (2) *create* a set *of* activities that will help ensure that every software engineering Work product exhibits high quality, (3) perform quality assurance activities on every software project, (4) use metrics to develop strategies to improving your software process and as a consequence the quality *of.* the end product.

Who does it? Everyone involved in the software engineering process is responsible for quality.

Why is it important? You can do it right, *or* you can do it over again. If a software team stresses quality in all software engineering activities, it reduces the amount of rework that it must do that results in lower costs, and more importantly, improved time-to-market.

What are the steps? Before software quality assurance activities can be initiated, it is important to define 'software quality' at a number of different levels of abstraction, Once you understand what quality is, a software team must identify a set of SQA activities that will filter errors out of work products before they are passed on.

What is the work product? A Software Quality Assurance Plan is created to define a software team's SQA strategy. During analysis, design, and code generation, the primary SQA work product is the formal technical review summary report. During testing, test plans and procedures are produced. Other work products associated with process improvement may also be generated.

How do I ensure that I've done it right? Find errors before they become defects! That is, work to improve your defect removal efficiency, thereby reducing the amount of rework that your software team has to perform.

SQA encompasses:

(1) A quality management approach
(2) Effective software engineering technology (methods and tools)
(3) Formal technical reviews that are applied throughout the software process
(4) A multi-tiered testing strategy

(5) Control of software documentation and the changes made to it
(6) A procedure to ensure compliance with software development standards (when applicable)
(7) Measurement and reporting mechanisms.

Software quality is defined as conformance to explicitly stated functional and performance requirements, documents, and standards. The factors that affect software quality are a complex combination of conditions that can be measured based on data, such as audit-ability, completeness, consistency, error tolerance, and expandability.1n addition, this data includes hardware independence, software system independence, modularity, security, and simplicity.

Software quality assurance is a planned and systematic approach necessary to ensure the quality of a software product. Software reviews filter the product of all errors. Software reviews can be conducted at all stages in the DLC of a software product, such as analysis, design, and coding.

Testing is an important element of SQA activity. There are various testing tools to automate testing process. The SQA plan is used as the template for all SQA activities planned for a software project and includes details of the SQA activities to be performed during project execution.

SCM is used to establish and maintain integrity of software items and ensure that they can be traced easily. SCM helps define a library structure for storage and retrieval of software items. SCM helps assess the impact of a recommended change and make decisions depending on the costs and benefits. SCM needs to be performed at all phases in the DLC of a software project.

The various SCM activities are identifying changes, controlling changes, controlling versions, implementing changes, and communicating changes. These activities are independent of the supervision of the project or the product manager. This is to ensure objectivity in SCM. The scope of SCM is not limited by code and includes requirements, design, database structures, test plans, and documentation. SCM procedures vary with the project.

10.2 Producing quality software

As we have seen, one of the main problems in producing quality software is the difficulty in determining the degree of quality within the software. As there is no single widely accepted definition for quality, and because different people perceive quality in different ways, both the developer and the customer must reach agreement on metrics for quality' (this is discussed in more detail later). The method of measuring quality may differ for different projects.

This problem is discussed in a paper by Wesselius and Ververs (1990), in which they conclude that complete objectivity in quality assessment cannot be achieved. They identify three distinct components of quality:

- An objectively assessable component
- A subjectively assessable component
- A non-assessable component

The quality of a product is objectively assessable when the characteristics of the product, as stated in the requirements specification, can be identified.

The quality of a product is subjectively assessable when the characteristics of the product comply with the customer's expectations.

The quality of a product is non-assessable when it behaves according to our expectations in situations that have not been foreseen.

Wesselius and Ververs suggest that, for the quality of a software product to be assessable, as many characteristics as possible should be moved from the subjective and non-assessable components to the assessable component. Essentially, this means that the requirements specification must describe as many measurable characteristics of the product as possible.

Experience supports Wesselius and Ververs' conclusions. Badly defined requirements are always a source of dispute between developer and customer. Well-defined, detailed and measurable requirements minimize disputes and disagreements when the development of the product is complete.

However, many development methods have a prolonged interval between the specification of requirements and the delivery of the product (refer to Chapter 4 for a discussion of the software development cycle). The determination of quality should not be postponed until development is complete. Effective software quality control requires frequent assessments throughout the development cycle. Thus, effective quality control coupled with a good requirements specification will clearly increase the quality of the final product.

10.3 Quality Control

Variation control may be equated to quality control. But how do we achieve quality control? Quality control involves the series of inspections, reviews, and tests used throughout the software process to ensure each work product meets the requirements placed upon it.

Quality control includes a feedback loop to the process that created the work product. The combination of measurement and feedback allows us to tune the

process when the work products created fail to meet their specifications. This approach views quality control as part of the manufacturing process.

Quality control activities may be fully automated, entirely manual, or a combination of automated tools and human interaction. A key concept of quality control is that all work products have defined; measurable specifications to which we may compare the output of each process. The feedback loop is essential to minimize the defects produced.

10.4 Quality Control Myths

The establishment of effective quality control frequently encounters various misconceptions and myths, the most common of which is related to the cost effectiveness of quality control. Cobb and Mills (1990) list several of these myths, and suggest methods of combating them. Two of the more prevalent myths identified by Cobb and Mills are described below.

Myth: Quality costs money. This is one of the most common myths (not only in software development). In fact, quality in software usually saves money. Poor quality breeds failure. There is a positive correlation between failures and cost in that it is more expensive to remove execution failures designed into software than to design software to exclude execution failures.

Myth: Software failures are unavoidable. This is one of the worst myths because the statement is partly true, and is therefore often used as an excuse to justify poor quality software. The claim that ‘there is always another bug' should never be a parameter in the design or implementation of software.

As these myths lose ground in modern approaches to software development, the demand for suitable quality control standards and procedures increases. The IEEE issued their first standard for software quality assurance plans in 1984 (IEEE .1984), followed by a detailed guide to support the standard, issued in 1986. The US Department of Defense issued a separate standard 2168 for defense systems software quality programs (DOD 1988b), which forms a companion to the famous US DOD standard 2167A (DOD 1988a) for defense systems software development. The European ISO standard 9000-3, of 1990 (ISO 1990) gives a broader meaning to the term quality assurance and covers configuration control too.

10.5 Resources for quality control

When the SQA mandate includes configuration control activities, the required resources will also include those required for configuration control. Merging SQA and configuration control is not uncommon, and can eliminate some duplication of assignments and activities. Two alternative organizational charts are shown in

Fig. 8.5. Note that for small projects, merging the two groups may mean simply assigning both responsibilities to the same person.

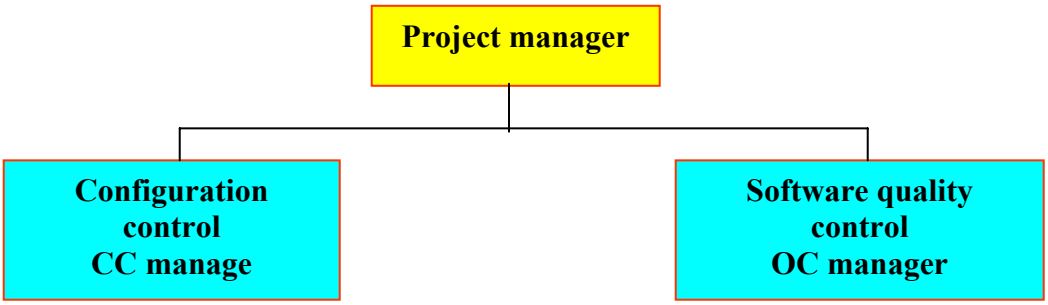
Though many tools are common to quality control and configuration control, few tools are specifically designed for quality control. The following are some of the general support tools that can be useful in supporting SQA activities:

- Documentation utilities
- Software design tools
- Debugging aids
- Structured preprocessors
- File comparators
- Structure analyzers
- Standards auditors
- Simulators
- Execution analyzers
- Performance monitors
- Statistical analysis packages
- Integrated CASE tools
- Test drivers
- Test case generators

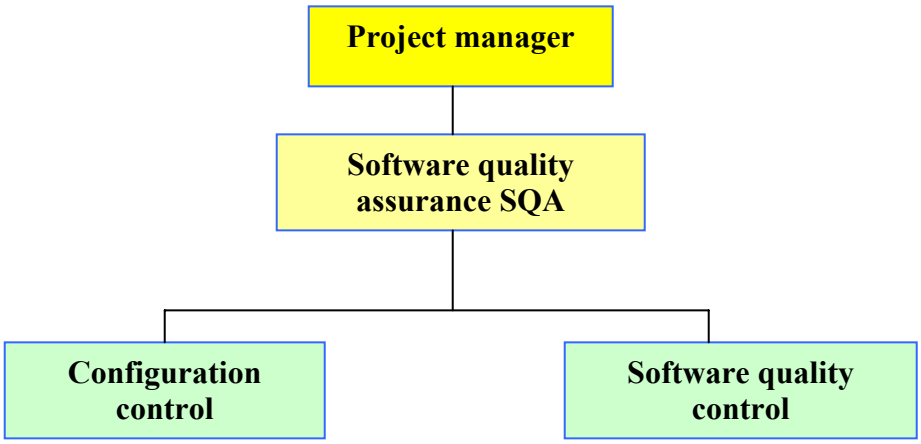
These tools support quality control in all phases of software development. Documentation aids can provide partially automatic document writers, spelling checkers and thesauruses etc. Structured preprocessors (such as the UNIX utility *lint)* are useful both to standardize code listings. And to provide additional compile-time warnings that compilers often overlook. Early warnings regarding possible execution time problems can be provided by simulators, execution time analyzers and performance monitors. Substantial software system testing can often be performed automatically by test suite generators and automatic test executors.

All SQA tools to be used during software development should be identified and described in the SQA plan. This plan includes a description of all required quality assurance resources and details of how they will be applied. Thus, at the start of the project SQA resources can be budgeted and procured as part of the required project development resources.

(a)



(b)



10.6 Quality Assurance

Quality assurance consists of the auditing and reporting functions of management. The goal of quality assurance is to provide management with the data necessary to I be informed about product quality, thereby gaining insight and confidence that product quality is meeting its goals. Of course, if the data provided through quality assurance identify problems, it is management's responsibility to address the problems, and apply the necessary resources to resolve quality issues.

Software Quality Assurance

Even the most jaded software developers will agree that high-quality software is an important goal. But how do we define quality? A wag once said, 'Every program does something right, it just may not be the thing that we want it to do.'

Many definitions of software quality have been proposed in the literature. For our purposes, *software quality* is defined as:

Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

There is little question that this definition could be modified or extended. In fact, a definitive definition of software quality could be debated endlessly. For the purposes of this book, the definition serves to emphasize three important points:

1. Software requirements are the foundation from which quality is measured. Lack of conformance to requirements is lack of quality.

2. Specified standards define a set of development criteria that guide the manner in which software is engineered. If the criteria are not followed, lack of quality will almost surely result.

3. A set of implicit requirements often goes unmentioned (e.g., the desire for ease of use and good maintainability). If software conforms to its explicit requirements but fails to meet implicit requirements, software quality is suspect.

Background Issues

Quality assurance is an essential activity for any business that produces products to be used by others. Prior to the twentieth century, quality assurance was the sole responsibility of the craftsperson who built a product. The first formal quality *assur*ance and control function was introduced at Bell Labs in 1916 and spread rapidly throughout the manufacturing world. During the 1940s, more formal approaches to quality control were suggested. These relied on measurement and continuous process improvement as key elements of quality management.

Today, every company has mechanisms to ensure quality in its products. In fact, explicit statements of a company's concern for quality have become a marketing ploy during the past few decades.

The history of quality assurance in software development parallels the history of quality in hardware manufacturing. During the early days of computing (1950s and 1960s), quality was the sole responsibility of the programmer. Standards for quality assurance for software were introduced in military contract software development during the 1970s and have spread rapidly into software development in the commercial world [IEE94].

Extending the definition presented earlier, software quality assurance is a planned and systematic pattern of actions [SCH98] that are required to ensure high quality in software.

The scope of quality assurance responsibility might best be characterized by paraphrasing a once-popular automobile commercial:

Quality is Job # 1'. The implication for software is that many different constituencies have software quality assurance responsibility-software engineers, project managers, customers, salespeople, and the individuals who serve within an SQA group.

The SQA group serves as the customer's in-house representative. That is, the people who perform SQA must look at the software from the customer's point of view. Has software development been conducted according to pre-established standards? Have the technical disciplines properly performed their roles as part of the SQA activity? The SQA group attempts to *answer* these and other questions to ensure that software quality is maintained.

SQA Activities

Software quality assurance is composed of a variety of tasks associated with two different constituencies-the software engineers who do technical work and an SQA group that has responsibility for quality assurance planning, oversight, record keeping, analysis, and reporting. 'If Software engineers address quality and perform quality assurance and quality control activities by applying solid technical methods and measures, conducting formal technical reviews, and performing well-planned software testing.

The charter of the SQA group is to assist the software team in achieving a high-quality end product. The Software Engineering Institute [PAU93] recommends a set of SQA activities that address quality assurance planning, oversight, record keeping, analysis, and reporting.

These activities are performed (or facilitated) by an independent SQA group that:

1. **Prepares SQA plan for a project**

   The plan is developed during project planning and is reviewed by all interested parties. Quality assurance activities performed by the software engineering team and the SQA group are governed by the plan. The plan identifies

- evaluations to be performed
- audits and reviews to be performed
- standards that are applicable to the project

- procedures for error reporting and tracking
- documents to be produced by the SQA group
- amount of feedback provided to the software project team

2. **Participates in the development of the project's software process description**

   The software team selects a process for the work to be performed. The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-900 I), and other parts of the software project plan.

3. **Reviews software engineering activities to verify compliance with the defined software process.** The SQA group identifies, documents, and tracks deviations from the process and verifies that corrections have been made.

4. **Audits designated software work products to verify compliance with those defined as part of the software process.** The SQA group reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made; and periodically reports the results of its work to the project manager.

5. **Ensures that deviations in software work and work products are documented and handled according to a documented procedure.** Deviations may be encountered in the project plan, process description, - applicable standards, or technical work products.

6. **Records any noncompliance and reports to senior management**. Noncompliance items are tracked until they are resolved.

   In addition to these activities, the SQA group coordinates the control and management of change and helps to collect and analyze software metrics.

10.7 SQA Plan

The SQA plan serves as the template for SQA activities planned for each software project. The SQA group and the project team develop the SQA plan. The initial two sections of the plan describe the purpose and references of the SQA plan. The next section records details of the roles and responsibilities for maintaining software product quality.

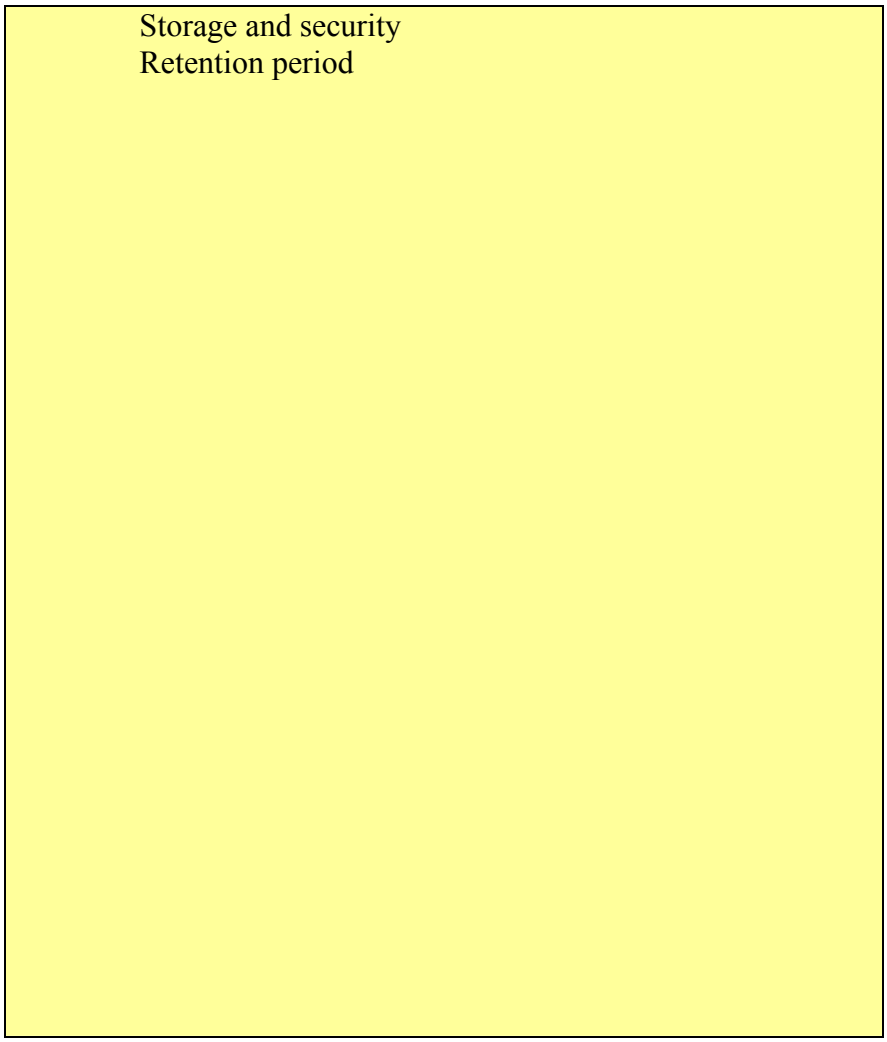
The Documentation section of the SQA plan describes each of the work products produced during the software process. This section defines the minimum set of work products that are acceptable to achieve high quality. The Documentation section consists of:

- Project document such as project plan, requirements document, test cases, test reports, user manual, and administrative manuals
- Models such as ERDs, class hierarchies
- Technical document such as specifications, test plans
- User document such as help files

All applicable standards to be used in the project are listed in the Standards and Guidelines section of the SQA plan. The standards and practices applied are the document standards, coding standards, and review guidelines.

Example of the contents of a software quality assurance plan

1. Software quality assurance organization and resources
Organization structure.
Personnel skill level and qualifications
Resources
2. SQA standards, procedures, policies and guidelines
3. SQA documentation requirements
List of all documentation subject to quality control
Description of method of evaluation and approval
4. SQA software requirements
Evaluation and approval of software
Description of method of evaluation
Evaluation of the software development process
Evaluation of reused software
Evaluation of non-deliverable software
5. Evaluation of storage, handling and delivery
Project documents
Software
Data files
6. Reviews and audits
7. Software configuration management (when not addressed in a separate document)
8. Problem reporting and corrective action
9. Evaluation of test procedures
10. Tools techniques and methodologies
11. Quality control of subcontractors, vendors and suppliers
12. Additional control
Miscellaneous control procedures
Project specific control
13. SQA reporting, records and documentation.
Status reporting procedures
Maintenance



10.8 Software quality metrics

Much attention has been devoted to questions associated with the measurement of quality. How do we determine the extent to which a software product contains this vague attribute called quality? When is the quality of a software product high and when is it low?

One of the more recent developments in quality assurance (not only for software) is the realization that quality is not a binary attribute that either exists or does not exist. Kaposi and Myers (1990), in a paper on measurement-based quality assurance, have stated their belief that 'the quality assurance of products and processes of software engineering *must be based on measurement* [7]. The earlier the measurement of quality begins, the earlier problems will be located. Cohen *et al.* (1986), in addressing the cost of removing errors during the early phases of software development, proclaim the existence of the famous *exponential law*[8].

The quality of two products can be compared, and it is perfectly acceptable to claim that the quality of one product is greater than the quality of another. It is also acceptable to measure quality and deduce the extent of expected faults based on the measured result.

The set of measurable values associated with the quality of a product is referred to as the product's quality metrics. Software quality metrics can be used to determine the extent to which a software product meets its requirements. The use of quality metrics increases the objectivity of the evaluation of product quality. Human evaluation of quality is subjective, and is therefore a possible source of disagreement, particularly between customer and developer.

A number of methods for establishing software quality metrics are currently being developed, though no generally accepted standard has yet emerged. For example, an initial draft of IEEE Std-1061 (1990) includes a detailed discussion of software quality metrics in general, including a suggested methodology for applying metrics, and many examples and guidelines. Quality metrics, once defined, do indeed increase objectivity, but the definition itself is not necessarily objective and greatly depends upon the needs of the organization that produces the definition.

The basic approach for applying software quality metrics requires:

- The identification of all required software quality attributes. This is usually derived from the software requirements specification.
- Determinations of measurable values to be associated with each quality attribute. A description of the method by which each measurable value will be measured.
- A procedure for documenting the results of measuring the quality of the software product.

A set of many values can be used to determine the overall quality of a software product. However, a single measure can be created to represent the overall quality of the software product. This requires:

- A weighted method for combining the measured quality attributes into a single measure of quality for the product.

Some examples of software metrics are:

Reliability: The percentage of time that the system is successfully operational (e.g. 23 out of 24 hours produces: 100 x (23/24) percent)

Recoverability: The amount of time it takes for the system to recover after failure (e.g. 1hour to reload from backups and 30 minutes to reinitialize the system)

User-friendliness: The amount of training time needed for a new user

The measurement of software quality should not be performed only at the end of a project. The degree of quality should be measured at regular intervals during development. Thus, any major reduction in the overall measure of quality should act as a warning for the project manager that collective action is required. High quality at the end of the project is achieved by assuring high quality throughout the development of the project.

10.9 Some general guidelines

The basic software quality assurance activities cover the review and approval of the development methodology, the software and documentation, and the supervision and approval of testing. Other SQA activities, such as the supervision of reviews, the selection and approval of development tools, or the administration of configuration control, depend on the way SQA is adapted to a specific project. The size of the project is usually the determining factor. The following guidelines discuss some of the parameters to be considered for different types of project when planning SQA.

- In small projects, many SQA activities can be performed by the project manager. This includes the organization and supervision of reviews and audits, the evaluation and selection of development tools, and the selection and application of standards.

- Test procedures and testing are always best when conducted by a separate independent team (discussed later). The decision on whether supervision of testing activities can be assigned to SQA depends on many factors, including the independence of the SQA team, the size of the project and the 'complexity of the project.

- When testing is performed by an independent test team, SQAS involvement will be minimal. In most other cases it will be the responsibility of the SQA team to plan and supervise the testing of the system.

- As a general guideline, it is usually undesirable for SQA to be performed by a member of the development team. However, small projects often cannot justify the cost of a dedicated SQA engineer. This problem can be solved by having a single SQA engineer responsible for two or three small projects (with each project funding its share of the SQA services).

One additional guideline is based on the conclusions of Wesselius and Ververs (1990) for the application of *effective* quality control:

- The ability to control software quality is directly linked to the quality of the software requirements specification. Quality control requires the unambiguous specification of as many of the required characteristics of the software product as possible.

Lecture # 43

11. Application Tools

11.1 Managing Tasks in Microsoft Project 2000

Project management is a highly specia1ized and professional task. The chances that a project can go haywire are very high because project management includes numerous activities, such as planning, resource allocation, risk management, and estimation. You can automate all these activities using a tool, *Microsoft Project 2000.* Activities such as project planning, resource and cost allocation, and project tracking can be done easily using the various user-friendly features of Microsoft Project 2000.

Microsoft Project 2000 is a popular project management tool that also enables you to schedule and track your project plan. In Microsoft Project*,* every unit of work, which consumes effort, time, and money, is called a *task.* You use Microsoft Project to specify tasks and the expected duration of each task. Using the task-related information, Microsoft Project prepares a project schedule. The schedule displays a plan consisting of the start and end dates of each task.

In addition, it plots a pictorial Gantt chart. The Gantt chart view depicts the extension of tasks across days along with the corresponding resources.

11.2 Creating Tasks

To manage a project efficiently, it is important to treat the project as a decomposable entity that can be analyzed and resolved. A problem first needs to be broken into units and then analyzed for solutions because it is always easier to focus on micro level units. Microsoft Project enables you to perform these tasks. It assumes that you have performed the WBS exercise for the project. The WBS exercise helps you to come up with indivisible units called *tasks.* Therefore, a task is an activity that has a defined start and end dates.

For example, XYZ Inc. has to develop an ERP project. They plan to complete the project in five phases: analysis, design, development, testing, and implementation. During the analysis phase, the WBS exercise is performed and a list of tasks is defined for each of the subsequent phases. Using Microsoft Project, you can create tasks for each phase, specify their duration, and assign resources to them. You can also set relationships between tasks to identify the successor and predecessor tasks.

11.3 Types of Tasks

Microsoft Project enables you to create three types of tasks:

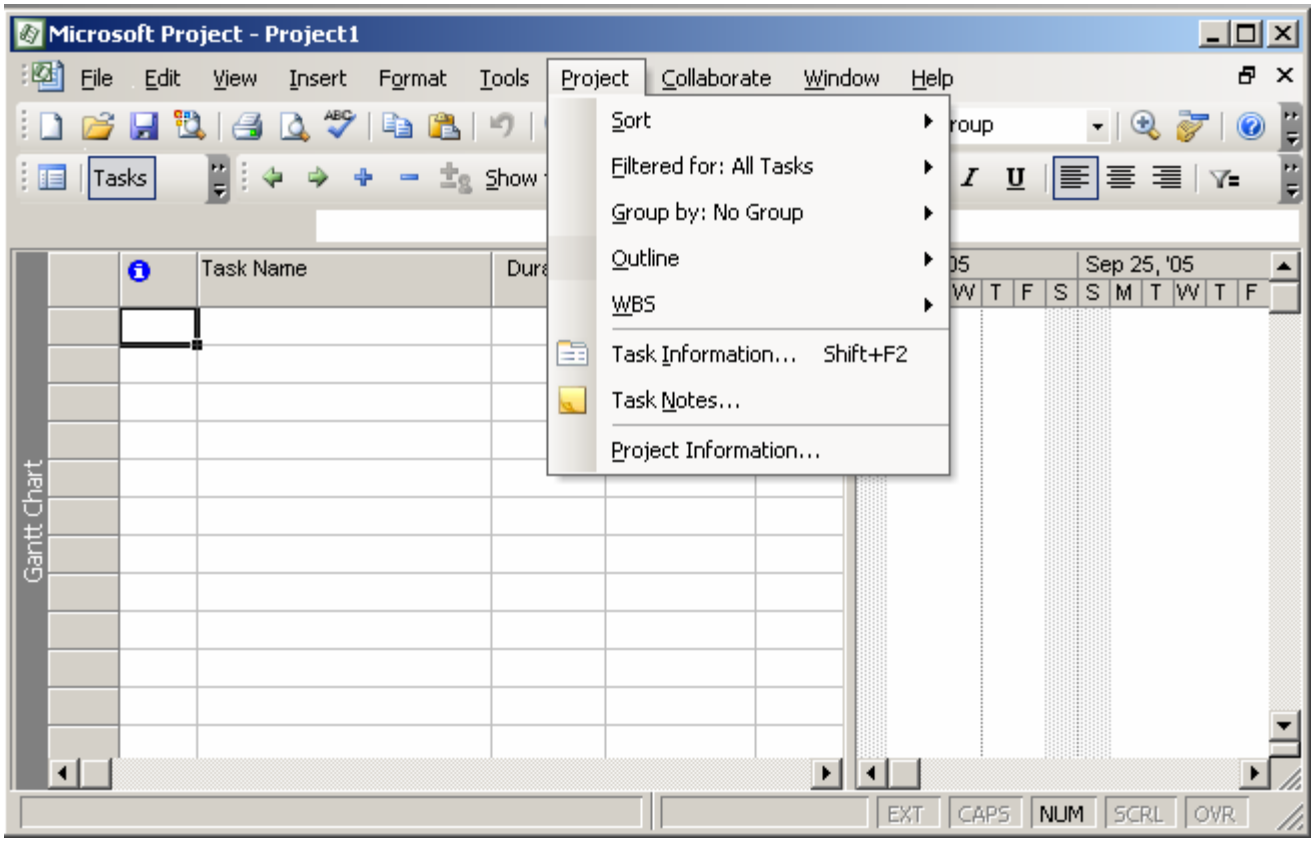
- General
- Milestone
- Recurring

General Tasks

General tasks are the usual tasks that are independent and have no special characteristics related to their duration or time. For example, Analysis is a general task that XYZ Inc. plans to conduct. To create the Analysis task, you can use the Project menu on the Microsoft Project worksheet. The steps to create the Analysis task are as follows.

1. Click the **Project** menu.

This displays the Project menu that contains all the task-related commands. Figure



11.1 displays the Project menu.

Figure 11.1: *Project Menu*

2. Click the Task information command on the Project menu.

   This displays the Task Information dialog box where you specify the details of the task being created.

   As shown in Figure 11.2, the Task Information dialog box contains five tabs: **General, Predecessors, Resources, Advanced, and Notes**. You use the General tab of the Task Information dialog box because you need to create a general task. The other tabs have specific purposes for each task. For example, you can specify the details for a task by using the **Advanced** tab. For the same task, you can use the Notes tab to add notes.

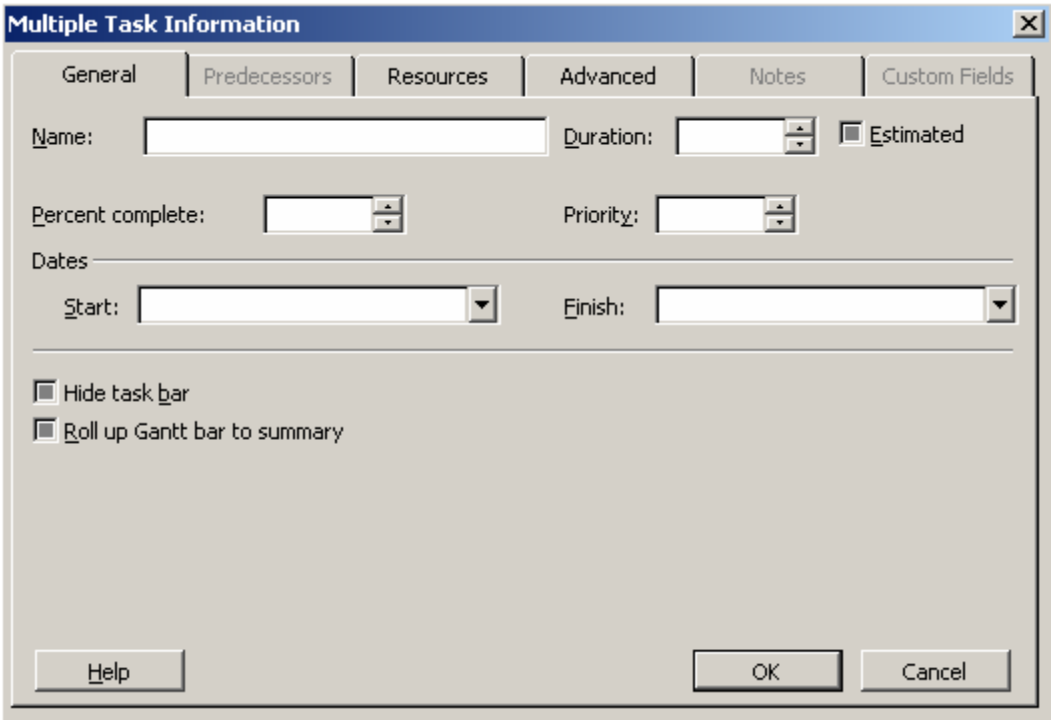


Figure 11.2: *Different Tabs in the Task Information Box*

Tip: In addition to the access key *I,* you can display the Task Information dialog box by using the Shift and F2 shortcut key combination.

3. Type **Analysis** in the **Name** text box.
4. Specify the duration of the Analysis task as **5d** in the **Duration** text box.

   The duration of a task signifies the period for which the task lasts in a project schedule. The start date is preset as the current date unless you enter a specific date. The finish date for the task is displayed as five days after the current date in the Gantt chart view. Duration can be specified in minutes, hours, days, weeks, months, or years.

If you are not sure of a confirmed duration, you can specify an estimated duration of a task.

Figure 11.3 displays the Task Information dialog box where task details are specified.

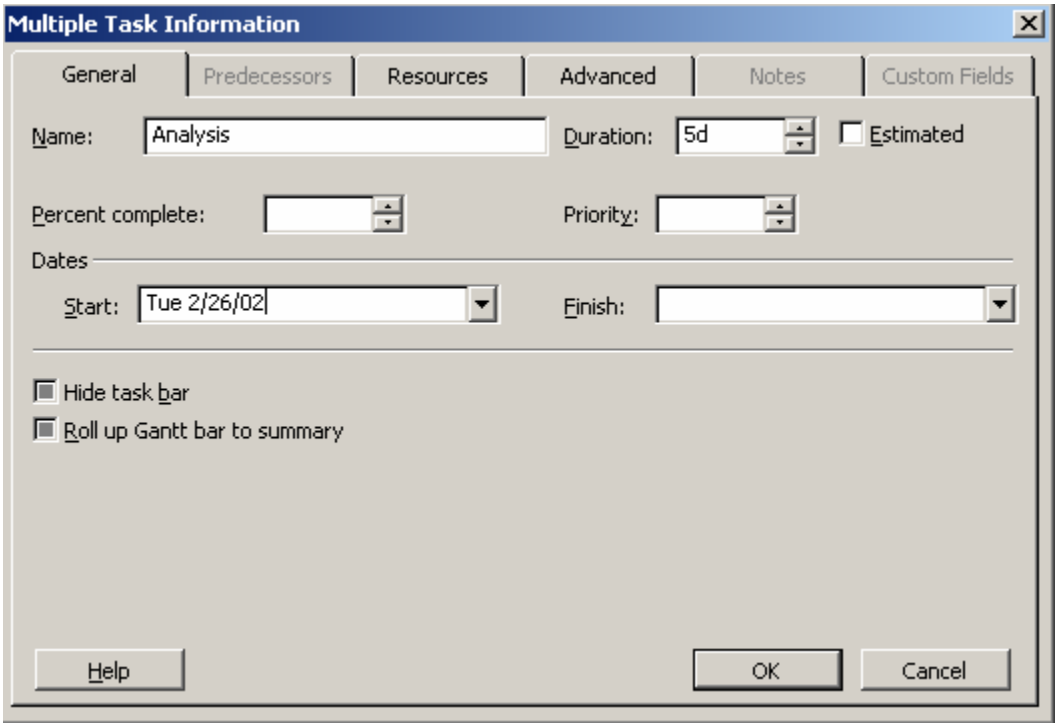


Figure 11.3: *Task Information Dialog Box*

Tip: To specify an estimated duration for a particular task, you can precede or succeed the duration with a question mark. Alternately, you can select the Estimated check box next to the Duration text box.

5. Click **OK** to confirm the details provided by you.

After a task is created, you; can specify the subtasks below it. For example, Project scope determination is a subtask of the Analysis task. To specify the subtask, you first create the Project scope determination subtask by using the Task information dialog box. Next, you specify a summary designation for the Analysis task. A summary designation indicates that the task represents a parent task for a group of related tasks below it. The start date of a summary task is the start date of the first subtask. Similarly, the finish date of the summary task is the finish date of the last subtask under the summary task.

To designate Analysis as the summary task, perform the following steps:

1. Create a task named **Project scope determination** using the **Project** menu.
   You need to create this task below the Analysis task.
2. Select the **Project scope** determination task.
3. Click the **Indent** icon on the Formatting toolbar.

The Indent icon is the green arrow pointing towards right. Clicking this icon indents the selected task to the right. Notice in Figure 11.4 that Microsoft Project formats the summary task in bold. Alternatively, you can use the left indent icon to make the Analysis task as a summary task.

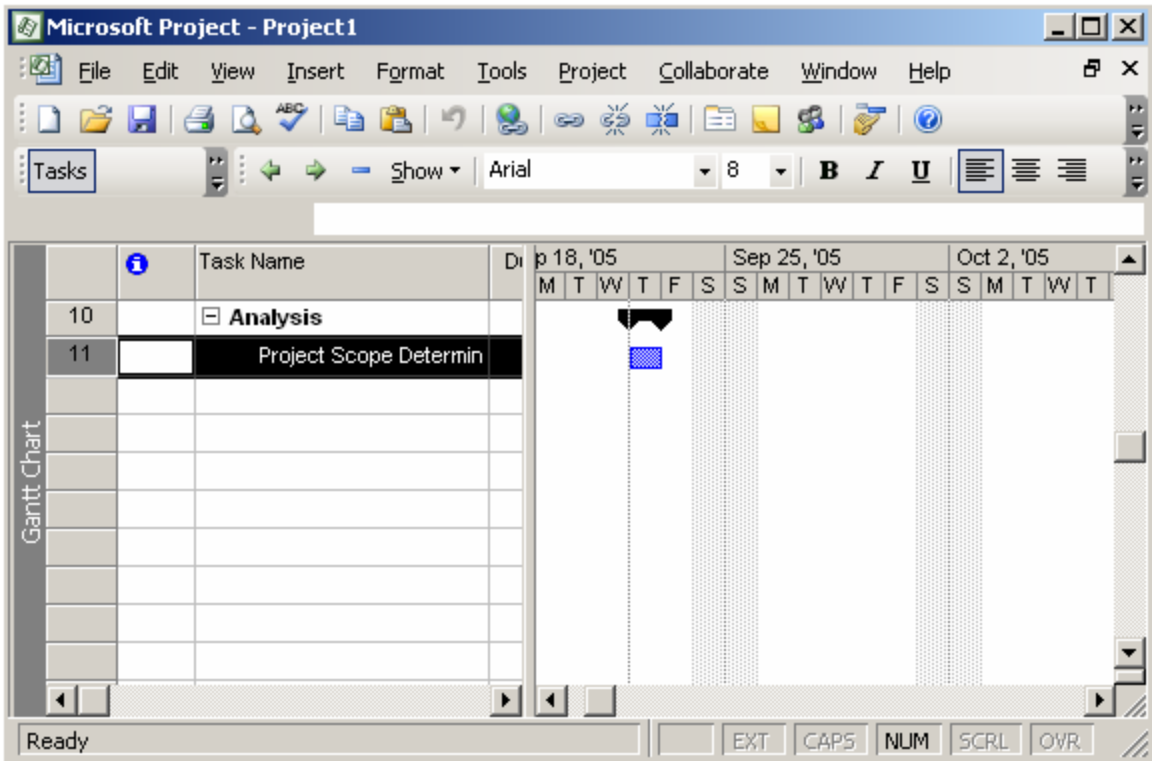


Figure 11.4: *Creating Summary Tasks*

Similarly, you create the rest of the tasks listed in the project plan of XYZ Inc. and specify subtasks below them. You store the list of tasks and their details in a file with an extension .mpp in Microsoft Project.

Milestone Tasks

Milestone tasks are the second type of tasks that you can create using Microsoft Project. These tasks have no duration. They serve as markers or reference nodes for significant accomplishments in a project. Examples of a milestone task are project startup, project end, pilot implemented, and SRS document sign off. Creating a milestone task is similar to creating any other task. The only difference is that after the task is created, you select an additional option named *Mark task as milestone* in the Task Information dialog box.

For example, in the analysis phase, XYZ Inc. has a milestone task named Analysis Complete. To indicate that Analysis Complete is a milestone task, perform the following steps.

1. Click the **Advanced** tab in the **Task information** dialog box.
   This tabbed page contains specialized options that enable you to create milestone tasks. It also contains options to specify constraints and deadlines.
2. Type **Analysis Complete** in the **Name** text box.
3. Select the **Mark task** as **milestone** check box.
   You do this to specify a task as a milestone task.
4. Click **OK** to close the **Task Information** dialog box.

Figure 11.5 displays the Mark task as milestone check box selected.

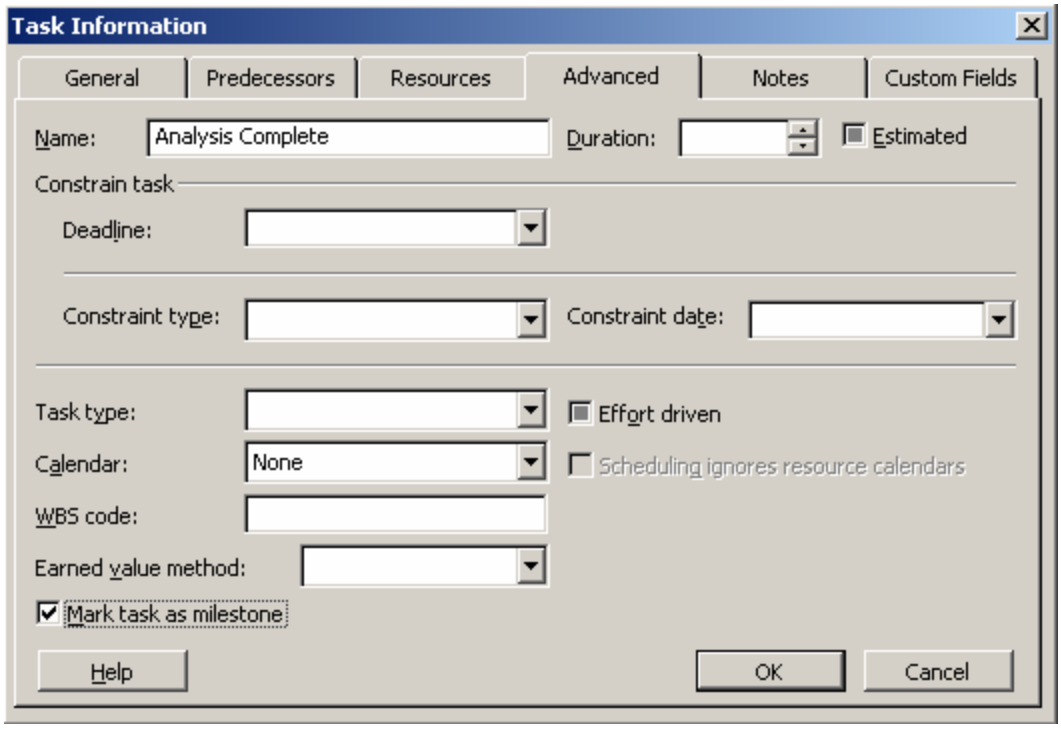


Figure 9.5: *Creating Milestone Tasks*

A milestone task is displayed as a task with no duration next to it in the Gantt chart view. Similarly, you create other milestone tasks in the Project file.

Recurring Tasks

Recurring tasks, as the name suggests, occur repeatedly at regular intervals. For example, a weekly project meeting with all the team members is a recurring task. You can schedule a recurring task in a project plan cycle as daily, weekly monthly, or yearly. You make this selection after deciding the seriousness of the

nature of the task. The seriousness of the nature of the task determines how often the task needs to recur.

For example, if it is a simple meeting to take stock of the situation, you can select the weekly option. Similarly, you can annually schedule a high level meeting of the board of directors. However, if an ongoing project is faltering due to frequent slippages, you need to closely monitor the activities of the project. To monitor the project activities effectively, you can convene a meeting every day.

You can also schedule the start and end dates of a recurring task. For example, for XYZ Ltd. you need to create the recurring task Project Meeting under the Analysis task. The Project Meeting task recurs every Friday until March 14, which is the project end date. You can create this task by using the Insert menu, and then specify the details in the Recurring Task Information dialog box. To create the recurring task you perform the following steps.

1. Select the task row where you want the task to appear.
2. Click the **Insert** menu.
3. Click the Recurring Task command to display the Recurring Task Information dialog box.
4. Type Project meeting in the Name text box.
5. Select the Friday check box in the Recurrence pattern group box. This step specifies that the task should recur every Friday. Note that the Weekly option button is selected by default.
6. Click the **End by** option button.

This ensures that the weekly task should continue only until February 14. Figure 11.6 displays this step.

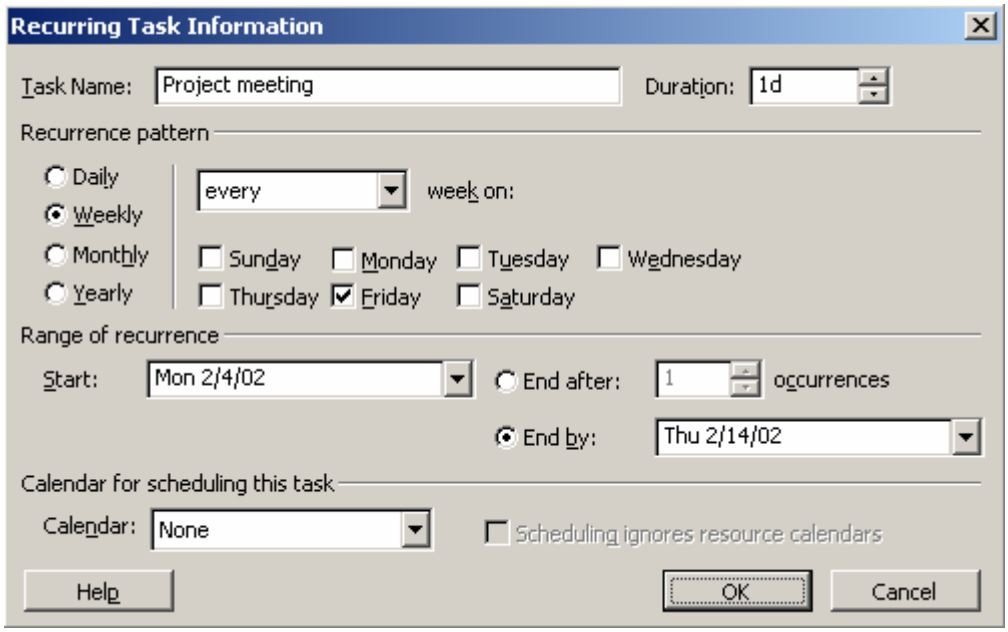


Figure 9.6: *Creating Recurring Tasks*

7. Select **February 14** from the calendar.
8. Click **OK** to close the **Recurring Task Information** dialog box.

A recurring task appears with a special circular arrow symbol in the indicator field. In the Gantt chart view, it appears at intervals on all the days it is scheduled until the last date specified.

11.4 Adding Constraints to Tasks

Constraints are defined as1illitations that negatively affect the schedule and quality of a project. Some of the typical project constraints are mentioned below:

- Duration of a project
- Resources of a Project
- Performance goals of a project

A constraint also affects other constraints. For example, you may need more resources if you reduce the project duration. You may also have to forgo some of the goals and features of the project if you decide to reduce the project duration. Therefore, it is important that you decide the tradeoff judiciously.

In Microsoft Project, you can specify constraints as limitations that are enforced on a task. For example, you can specify that a task must start on a particular date or not finish later than a particular date. Therefore, schedule-related constraints are available in Microsoft Project.

- As soon as possible (ASAP)
- As late as possible (ALAP)
- Start no later than (SNLT)
- Finish on later than (FNLT)
- Start no earlier than (SNET)
- Finish no earlier than (FNET)
- Must start on (MSO)
- Must finish on (MFO)

ASAP
When you use this constraint, Microsoft Project schedules a specific task as early as possible. For this reason you do not mention a specific date for such a task. Microsoft Project automatically assigns this constraint when you schedule a new task from the start date of a project. To change this constraint to a specific constraint, you can assign a specific start date to it.

ALAP
When you use this constraint, Microsoft Project schedules a specific task as late as possible. This is the reason why no specific date is mentioned for such a task.

Microsoft Project automatically assigns this constraint when you schedule a new task from the finish date of a project. To change this constraint to a specific constraint, you can assign a specific finish date to it.

SNLT
You select this constraint to specify the latest possible start date of a task. For example, you want to conduct a training program on quality standards for the development team. You want to conduct the training program before the development phase begins so that you have enough time on hand. Therefore, you select the SNLT constraint and specify the start date that is before the start of the development phase. This means the training begins on or before the specified date. You also keep in mind the duration of the training. Suppose it is a two-day training program and the project development start date is February 15. Therefore, after selecting the SNLT constraint, you will enter the start date of the training as February 13.

FNLT
You select this constraint to specify the latest possible date by which a task can be completed. For example, you want the project proposal task to be completed on or before the analysis phase, which is scheduled to start from February 20. Consequently, by enforcing this constraint on the project proposal task, you ensure that the finish date for this task is the same as the start date for the analysis phase.

SNET
This constraint enables you to schedule the start date of a task as early as possible. This is different from the ASAP constraint because the ASAP is a vague constraint with no specific start dates for a task. However, by choosing the SNET constraint, you can specify an early start date for a task. For example, the task to create test cases is dependent upon the construction and the unit testing phases. Therefore, the task to create test cases cannot begin unless the earlier two tasks are over. In this case, you would enforce the SNET constraint on the creation of test cases and specify a start date. This start date cannot be before the end date of the preceding tasks. Consequently, you can create test cases any time on or after this start date but not before.

FNET
You select this constraint when you do not want a task to be complete before a scheduled finish date. For example, during the unit-testing task you discover there were many bugs that were addressed by the developers. This makes you skeptical about the quality of the product. Therefore, you want to extend the finish date of unit testing and enforce the constraint FNET on it. Consequently, you will not be able to finish the unit-testing task before the specified date.

MSO
This constraint indicates that you must begin a task only on the specified start date and not later or before that. This is an inflexible constraint. It is usually enforced during a short cycle duration project when you are unsure whether each task mayor may not start on time. This may disturb the planned schedule of the entire project. You can enforce the MSO constraint for critical projects depending upon client requirements.

MFO
If you use the MSO constraint, it is most likely that you would use the MFO constraint also. Using the MFO constraint enables you to focus on completing a particular task. It is useless if you enforce a strict start date with no strict finish date for completing a task. You can again enforce this constraint for critical projects depending upon client requirements.

You enforce or use a constraint on a task by specifying the details in the advanced page of the Task Information dialog box. In that dialog box, you can also specify the start and the end dates of a constraint depending upon the type of constraint that you choose.

For example, for XYZ Inc. you need to add a constraint to the Develop test plans task. This task is a subtask of the Testing task. After creating both the tasks, you specify the duration of the Develop test plans task. You want that the task should not carry on for more than two days. You also need to impose a constraint that the task should not finish later than August 10. However, the management would appreciate if the task finishes before that.

To add the constraint to the Develop test plans task, you use the constraints feature in the Task Information dialog box. To access the constraints feature, you perform the following steps:

1. Click the **Advanced** tab in the **Task Information** dialog box.
2. Type **Develop test plans** in the Name text box of the **Advanced** tabbed page.
3. Type **2** in the **Duration** text box
   This specifies the maximum time allotted to complete the task. 4. Click the Constraint type box.
4. Click the **Constraint type** box.
5. Select the **Finish No Later Than** option in the displayed list.
6. Click the **Constraint date** list.
7. Select **August l0** as the finish date from the calendar that is displayed.
   Figure 9.7 displays this step.

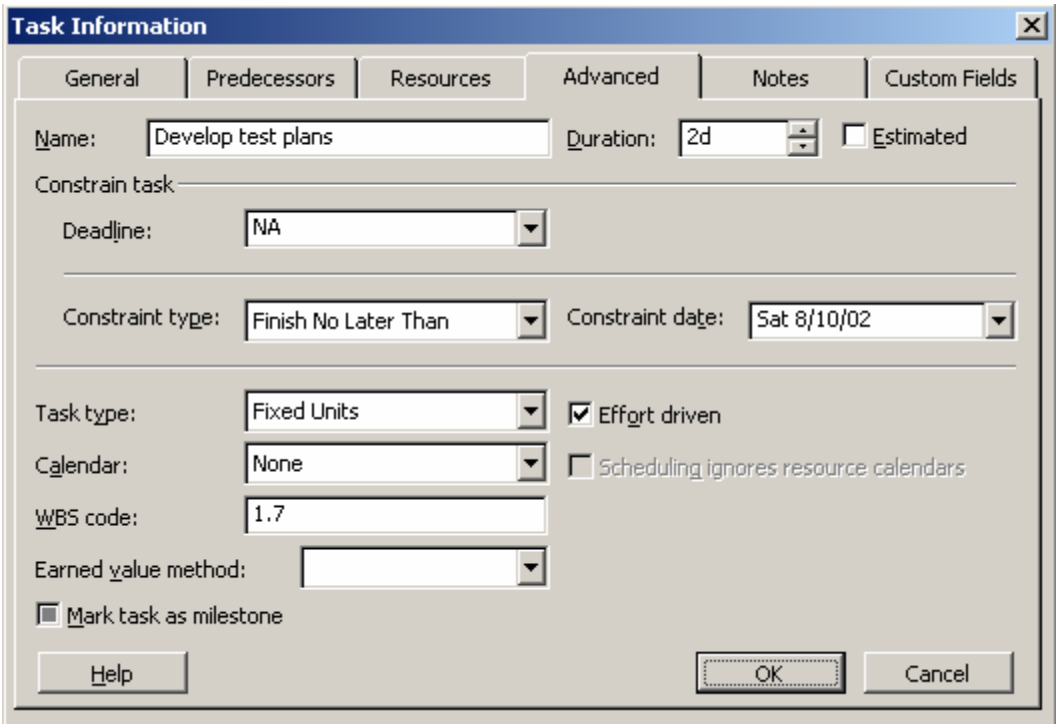


Figure 9.7: *Adding Constraints*

8. Click **OK** to close the **Task Information** dialog box.
   The Develop test plans task is displayed with an icon for Finish No Later Than constraint in the Task sheet of the Gantt chart view.

   Tip:
   After specifying a constraint type and its date, it is a good idea to specify the details of choosing a particular constraint type and its date. Subsequently, you may want to share this information with your colleagues in meetings. This can be done by clicking the Notes tab in the Task Information dialog box and entering your comments.

11.5 Adding Deadlines to Tasks

Deadlines are strict finish dates for a particular task. Microsoft Project considers a deadline to be a target date to complete a task. You add a deadline to a specific task if you track the progress of that task. Setting realistic deadlines to tasks also enables you to schedule the subsequent tasks.

Deadlines are important because they depict the finish date of a task. Therefore, you can track whether a task has exceeded the planned time or is completed before time. If the task is taking longer than expected, you can take remedial measures such as adding resources. Adding deadlines does not affect project schedule nor does it enforce a date constraint. However, it marks a task that has crossed its deadline with an indication mark next to it.

In the .mpp file that you are creating for XYZ Inc., you need to set July 15 as the deadline for the task Assign development staff; which is a subtask of the Development task. To set the deadline, you perform the following steps:

1. Create the task **Assign development staff** by using the **Project** menu.
2. Select the **Assign development staff** task.
   This is the task for which you need to set a deadline.
3. Click the **Advanced** tab in the **Task Information** dialog box.
4. Click the **Deadline** list box in the Advanced page.
   This is done to specify the deadline for the Assign development staff task.
5. Select **July 15** in the calendar to specify the deadline date.
6. Click **OK** to close the **Task Information** dialog box.

Similarly, you can add deadlines for the other tasks for the XYZ Inc. project.

11.6 Setting Dependencies between Tasks

There is dependency between tasks when a particular task cannot begin until its preceding task is completed. For example, the creation of the high-level design of a project cannot start until the client signs off the SRS document. The SRS document serves as a prerequisite for the immediate subsequent phase.

In Microsoft Project, you can specify four types of dependencies:

- Finish-to-Start (FS)
- Start-to-Start (SS)
- Finish-to-Finish (FF)
- Start-to-Finish (SF)

FS

In dependency relationships of tasks, the name of the relationship consists of the start or finish dates of the predecessor task and the start or finish date of the successor task. For example, the relationship between the task to finish the SRS document and the task to begin the high level design is FS. This is because you link the finish status of the predecessor task to the start status of the successor task. FS is the default relationship when you link any two tasks.

SS

In addition to FS, you can also begin two tasks at the same time. For example, you can begin the task to do functional testing and integration testing of the testing phase simultaneously. The dependency in a SS relationship occurs when both the predecessor and the successor tasks start at the same time. This is because these tasks do not share any common component nor are they dependent on each other. Therefore, the link between them is SS.

FF

You can also begin two tasks that finish simultaneously. In such a case, the dependency relationship between the two tasks is called an FF relationship. For example, the task to prepare a checklist for implementing a software product cannot end unless implementation testing for that software product is complete.

SF

Finally, you can have a SF relationship between tasks. This relationship indicates that a task cannot end until its preceding task starts. This relationship usually takes place when a project is scheduled from the project finish date. This relationship helps to minimize the risk of delay of a predecessor task if the successor task is dependent on the start of the predecessor task. For example, the testing phase can only start after the coding phase is complete.

11.7 Lag and Lead Time between Tasks

While adding dependencies to tasks, you can also specify lag and lead times for any two interdependent tasks. Lag time is defined as the excess time that is allowed to lapse between predecessor and the successor tasks. This results in a delay in the start of the successor task. For example, you may use this option if you feel that the tasks of the analysis phase are taking more than the planned time to complete. Consequently, you might not want to start the high--level design task as planned but postpone it. In this case, you would calculate the maximum lag time for the high-level design to start.

You specify the lag time for a task as a positive value in the Lag field cell of the Predecessors page. This page exists in the Task Information dialog box.

Lead time is defined as the overlap that exists between two dependent tasks. For example, you could decide to start testing units when developers have partially completed creating the applications for a project. Therefore, you move ahead of the schedule of a successor task before time.

You specify the lead time for a task as a negative value in the same Lag field cell of the Predecessors page. This page exists in the Task Information dialog box.

Consider an example to link two tasks, procuring resources and drafting a project plan. According to the plan, securing resources is a predecessor task while drifting a project plan is a successor task. You need to link the two tasks using the start-to-finish relationship because it is anticipated that the task to draft a plan cannot start before the resources are procured and incorporated in the project plan. During the course of the project, you feel that the successor task cannot begin as planned because procuring resources is too time-consuming. You analyze and deduce that the project plan can only be drafted four days after resources are procured.

You link the two tasks by using the Edit menu. After linking the two tasks, you set the type of relationship as start-to finish. Assume that for the XYZ Inc:

project, you need to link the Develop code and Developer primary testing tasks. You have created the Development summary task before. Now, you create these two tasks under the Development summary task. Then, you set a relationship between the tasks. Finally, you set a four-day lag time between the two tasks. To link tasks and set the lag time between them, perform the following steps:

1. Create a task named **Develop code** by using the **Project** menu.
2. Create a task named **Developer primary testing** below the **Develop code** task by using the **Project** menu.
3. Select Develop code.
4. Select Developer **primary testing** with the **Ctrl** key pressed.
5. Click the **Edit** menu.
6. Click the **Link Tasks** command.

The two tasks are displayed as linked to each other in the Gantt chart view as you see in Figure 11.8.

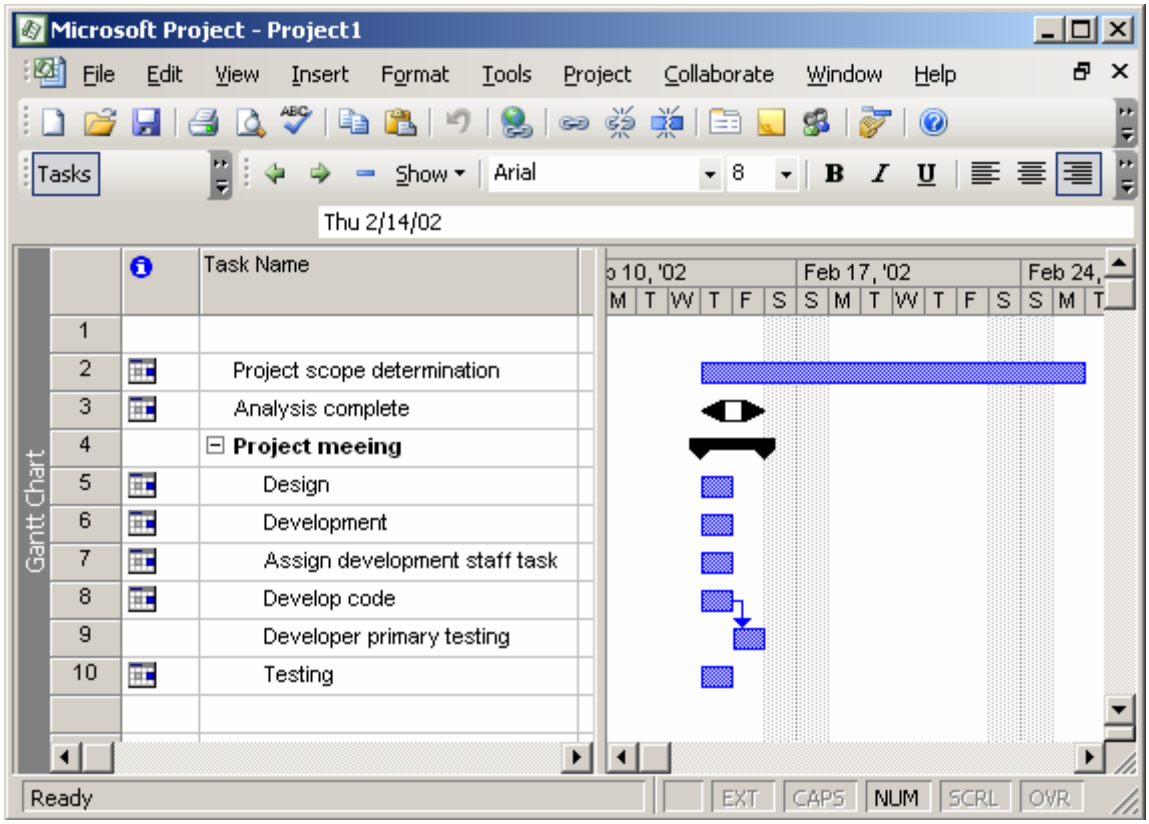


Figure 11.8: *Linking Tasks*

7. To establish the start-to-finish, relationship between the two tasks, select the **Developer primary testing** task and click the **Project** menu.
8. Click the **Task Information** command.

This information displays the Task Information dialog box wherein you can establish a defined relationship between the two tasks by using the Predecessors tab.

9. Click the **Predecessors** tab in the Task Information dialog box.
10. Select the **Start-to-Finish (SF)** relationship in the **Type** field cell.

This defines a start-to-finish relationship for the tasks.

11. Type **4d** in the **Lag** field cell to specify four days lag time between the two tasks. This step is displayed in Figure 11.9.

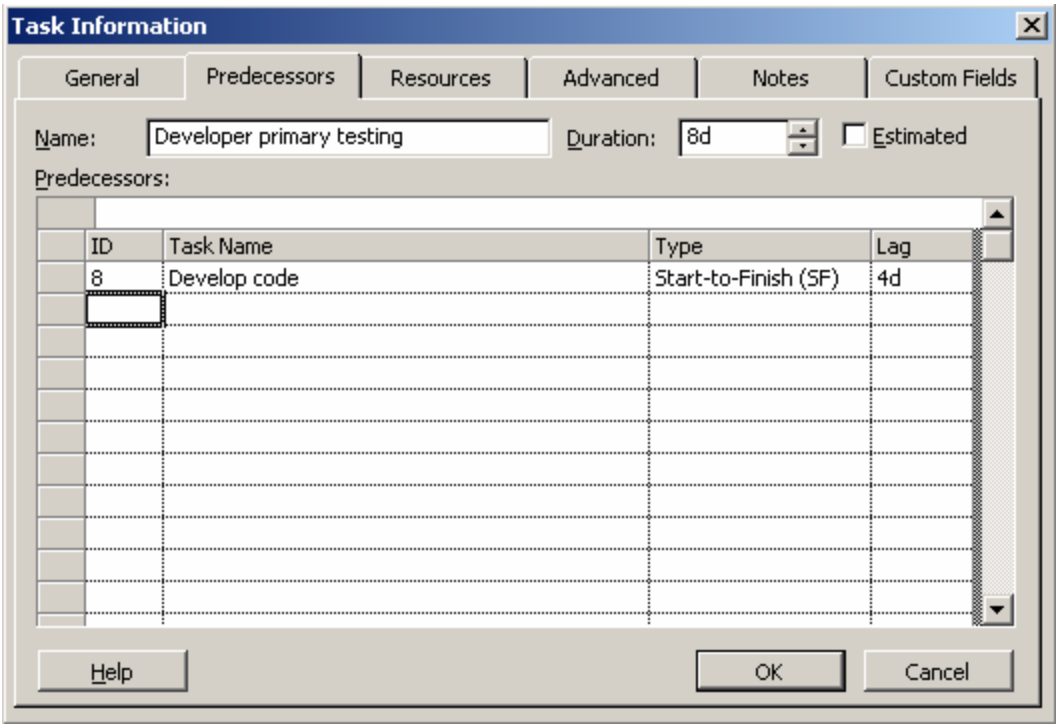


Figure 11.9: *Specifying Value for the Lag Field*

12. Click **OK** to close the dialog box.

The tasks Develop code and Developer primary testing appear linked to each other with the symbol of Start-to-Finish relationship in the Gantt chart view.

11.8 Splitting Tasks

A split in a task is defined as an abrupt and sudden interruption in the planned schedule. The need to split tasks arises because practically no task can be completed without a break. Due to resource crunch, a planned task is normally halted for some time.

For example, at XYZ Inc., both the tasks to train developers and to test a software module require a Windows 2000 Server. However, depending upon the criticality of the tasks, you can split one of the non critical tasks. In this case, you decide to stop testing the software module and continue training the developers. To specify the task using Microsoft Project, you first create a Testing task as the summary task. After creating the Testing task, you create the Unit testing task under the Testing task. Next, you split the Unit testing task.

To split the Unit testing task, you perform the following steps:

1. Create a task named **Unit testing** below the Testing summary task.
2. Select the **Unit testing** task.
3. Click the **Edit** menu.
   The Edit menu contains the command that enables you to split a task.
4. Click the **Split Task** command on the Edit menu.
   Upon clicking the Split Task command, the shape of the cursor changes. In addition, a small yellow box with a tool tip suggesting how to split a task hovers on the task view of the Microsoft Project window.
5. Drag the changed mouse cursor over the task in the Gantt chart view.

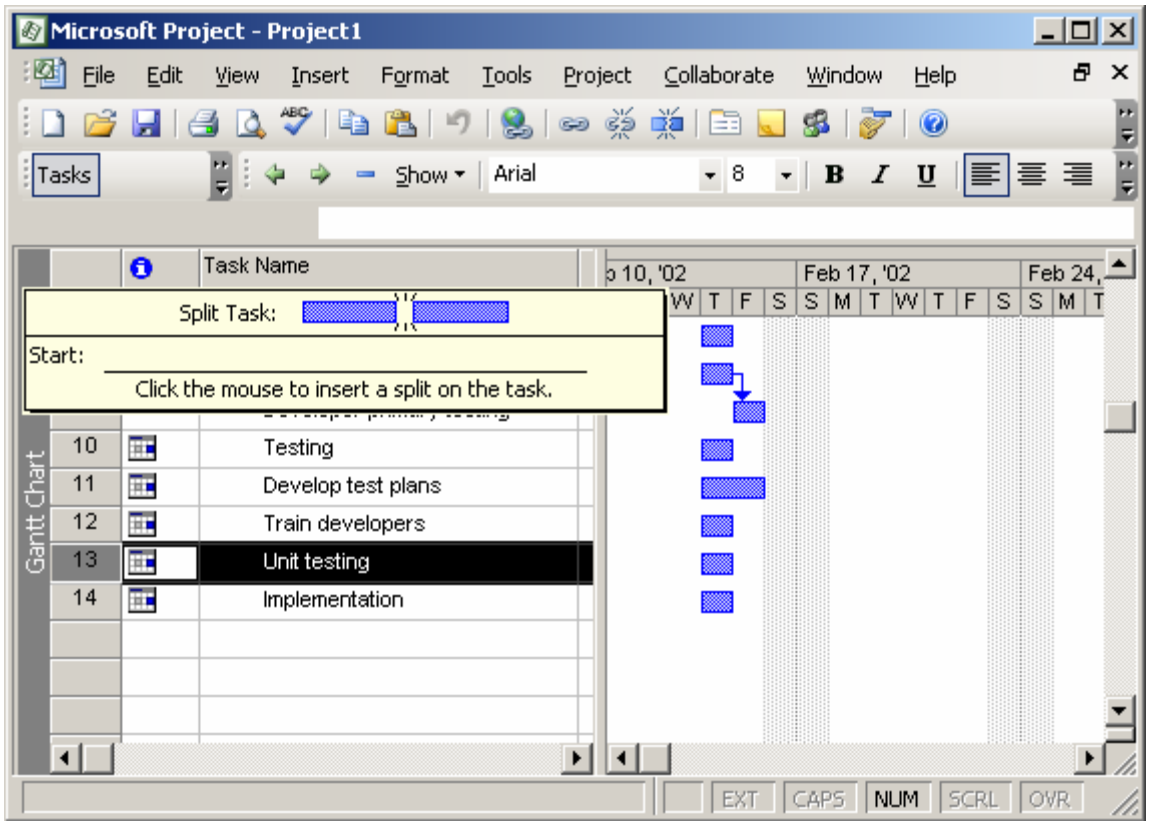


Figure 11.10: *Splitting Tasks*

In Figure 11.10, you can see the task being split into the required number of days. To split a task for a specified number of days, you drag the mouse pointer to a particular date. For example, a task is scheduled to start from May 5. You want a part of the task to start again from May 15. Therefore, there is a gap between May 5 and May 15. To start the task again from May 15, you drag the last end of the task and release the mouse pointer on May 15 in the Gant Chart view.

11.9 Assigning Resources and Resource Costs to Tasks

Resources are means that help convert input to a task into output for the task. Resources are assigned to tasks for the following reasons:

- To track the costs of each resource
- To track the work completed and the material resources used in a project

There are two types of resources, work and material. Work resources refer, to the people and equipment. At the end of a project, these resources are not consumed. Unlike work resources, material resources are used completely by the end of the project. For example, to complete the task of writing the code for a module, you need human and material resources. You need material resources such as hardware, software, and electricity.

Assigning resources to a task is a two-step process. First, you add the available resources to a project. You can do this by adding resources by using the View menu and then clicking the Resource Sheet command on the View menu. When you click the Resource Sheet command, Microsoft Project changes the default Gantt chart view to the Resource Sheet view. Figure 11.11 displays the Resource Sheet view.

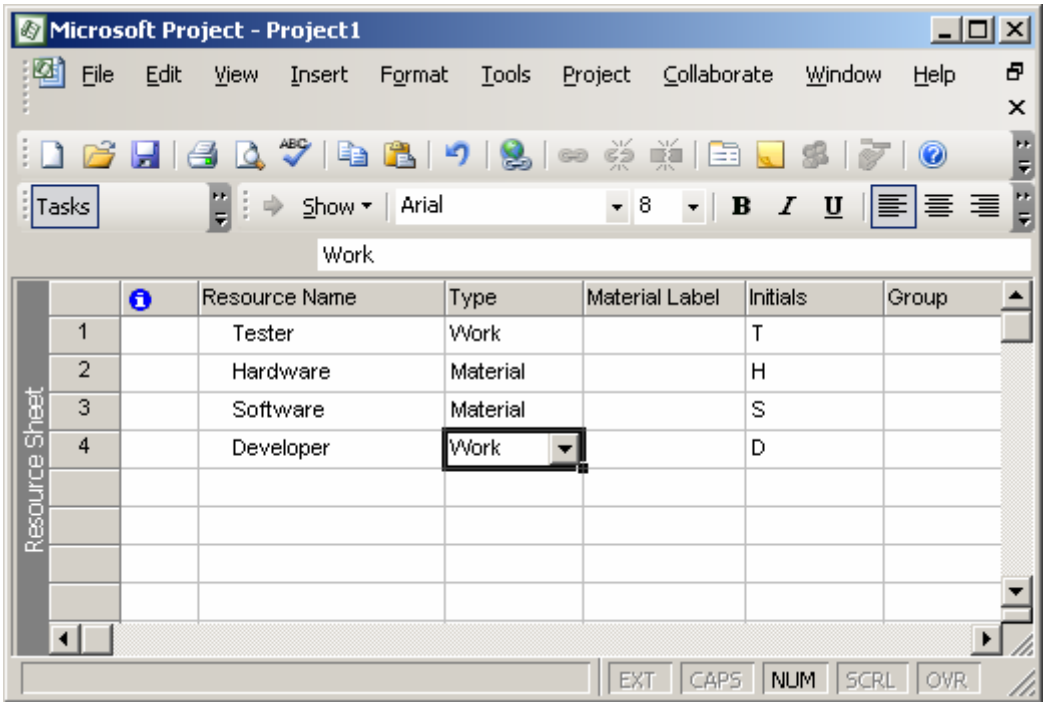


Figure 11.11: *Resource Sheet View*

Suppose for the XYZ Inc. project, you want to add a hardware resource in the Develop prototype task. The steps to add the resource are as follows:

1. Create a task named Develop prototype below the Analysis summary task.
2. Select the Develop prototype task.
3. Click the Tools menu.
4. The Tools menu contains the command to add a resource.
5. Point to the Resources command.
6. When you point to the Resources command, the Assign Resources command of the Resources command of the Resources submenu is displayed.
7. Click the **Assign** Resources command. Figure 11.12 displays the Assign Resources dialog box.

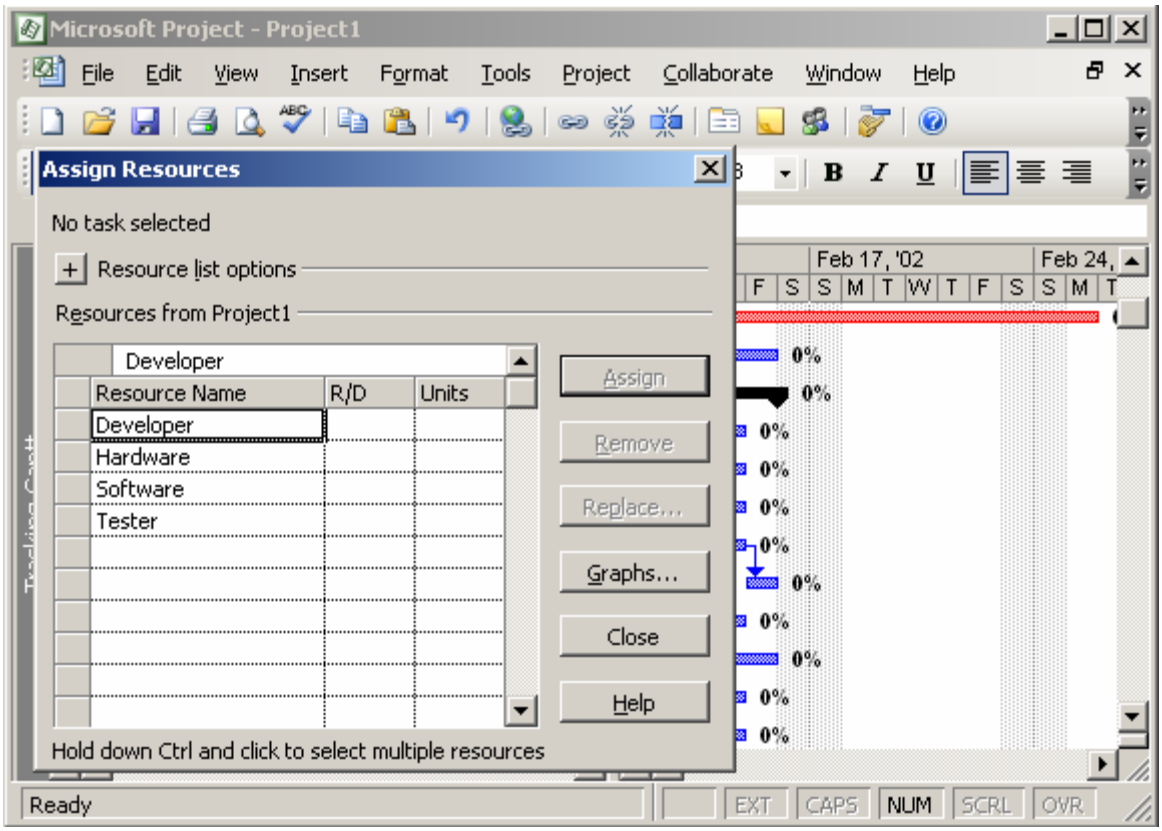


Figure 11.12: *Assigning Resources to Tasks*

The Assign Resources dialog box contains two fields, Name and Units. The Name field indicates the name of the available resources in the resource pool or a project. The Units field indicates the number of resources that can be used for a particular resource.

You can use the Assign Resources dialog box to assign units for both work and material resources. The unit for work resources is specified in terms of percentage. However, the unit for material is specified as a numeric value.

For example, you may select three computers or five workstations for the development team of a project. These values are not more than the predetermined value set for a resource in the Max.Units field of every resource in the Resource Sheet view.

Select the hardware resource for the Develop prototype task. You close the Assign Resources dialog box by clicking the Close button. The hardware resource is added next to the Develop prototype task.

11.10 Assigning Costs to Tasks

Cost is an important aspect of project scheduling. Controlling costs ensures the optimal use of resources. For example, to complete a specific task in a project plan, you can assign more resources to that task. Alternatively, you can increase the duration of that task depending on the cost of the resources that are used by that task.

You can assign two rates for any type of a resource. The rate for work resources are calculated on an hourly basis. However, the rate at which the materia1 resources are calculated can be:

- Standard rate: The normal rate at which a resource is procured
- Overtime rate: The rate that you pay if you overuse a work resource

For example, for the XYZ Inc. project, you need to specify the standard and the overtime rates for a work resource named Tester. To do this, you first assign the work resource Tester to the Unit Testing task that you created earlier. After assigning the resource to the Unit Testing task, perform the following steps to assign rates to the resource:

1. Click the **Resource Sheet** view of Microsoft Project
   The Resource Sheet view is used to display all resources and their respective costs.
2. Type Tester in the first cell of the Resource Name column
3. Select Tester.
   You do this because that is the work resource for which you need to specify the standard and overtime rates.
4. Click the Project menu to display the Resource Information command.
5. Click the Resource Information command on the Project menu.
6. Click the Cost tab in the dialog box.
   The costs tab provides information about the standard and overtime rates and the cost per unit of a particular resource.
7. Type $30.00/h in the first cell of the **Standard Rate** column as shown in Figure 9.13.
8. Type $10.00/h in the first cell of the **Overtime Rate** column.

The rates can also be entered as negative values to denote the decrease in rate as compared, to previous rates or as positive values to denote a comparative increase in rate, such as +20% or -20%.

9. Click OK to close the Resource **Information** dialog box.

While calculating costs for work resources; Microsoft Project calculates the resource cost by the hourly rates that you specify and the time required for completing a task. However, when you assign a material resource to a task, Microsoft Project automatically calculates the total cost of the resource by using the material resource rate specified by you and the quantity of material required for completing the task.

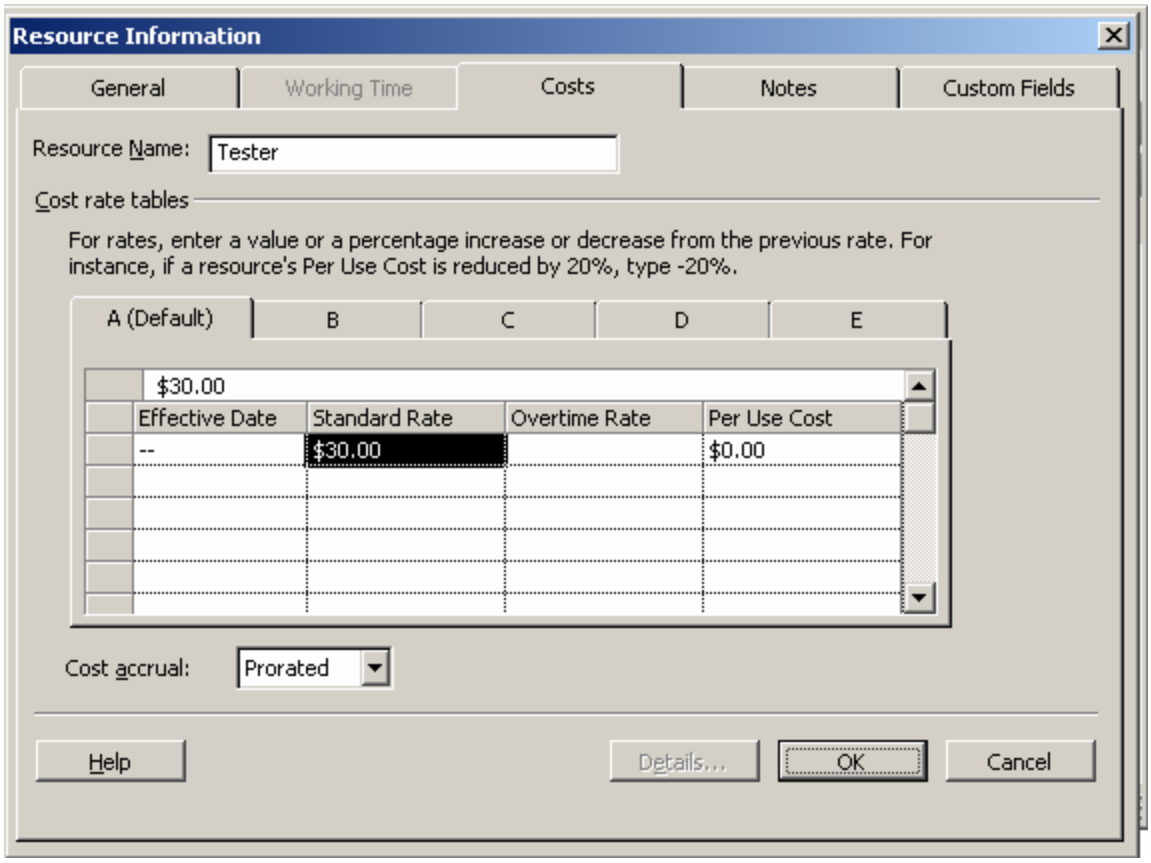


Figure 11.13: *Assigning Costs to Resources*

11.11 Tracking a Project Plan

Project management is a multiple-phase undertaking. The first phase is the project initiation or the project planning phase. This phase is followed by the project-tracking phase. Finally, you wrap up a project with the-project-end phase. Project tracking is an extremely in important activity in an organization. Unless a project is tracked effectively, it might never come close to termination. Project tracking also ensures strict adherence to project plans, requirements, and schedules.

Progress Lines

To track a project plan, you first view the progress of a project. You can view the progress of a project by applying progress lines to the tasks in the project. You can display progress lines in three ways,

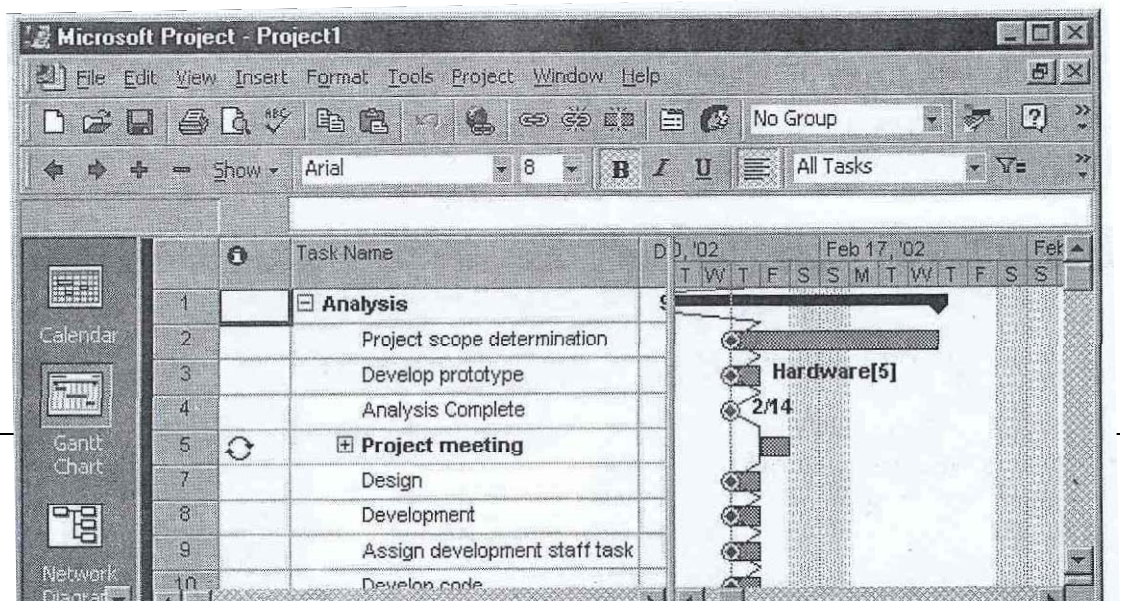
You can choose to display progress lines always at the current status of a project. To do this, you need to select the Always display current progress lines check box in the Progress Lines dialog box. Selecting this option displays progress lines every time based on the current status of a project.

The second way to display project lines is to display them on selected dates of a project plan. For example, you use this option when you need to know the progress status of a project on the jay development is completed. In this case, you select the Display selected progress lines check box to display progress lines on that day. Similarly, you can use this option to display progress lines on the day when testing is completed or the project-end date.

Finally, you can display progress lines at regular intervals such as daily, weekly, or monthly.1f you choose daily, you can further define its preciseness by choosing everyday, every second, or every third day, and so on. Alternatively, if you choose monthly, you can refine your choice by selecting the month and day when a progress line is displayed. To display progress lines for all the tasks listed in the Sample-Project.mpp file every third day, you perform the following steps:

1. Click the **Tools** menu
2. Point to the **Tracking** command to view different options for creating progress lines
3. On the Tracking submenu, click the **Progress Lines** command
4. The Progress Lines dialog box is displayed. In the dialog box, select the **Display progress lines at recurring intervals** check box.
5. After selecting the **Display progress lines at recurring intervals** check box, select the **Daily** option button.
6. To specify that the progress lines must be displayed every third day, click the: every box and select the **every 3rd** option in the list.
7. You can further choose to display the progress lines every third day of a week on a workday. This is shown in Figure 11.14
8. To confirm the specifications and close the **Progress Lines** dialog box, click **OK**.

Figure 11.14: *Setting Options for Progress Lines*



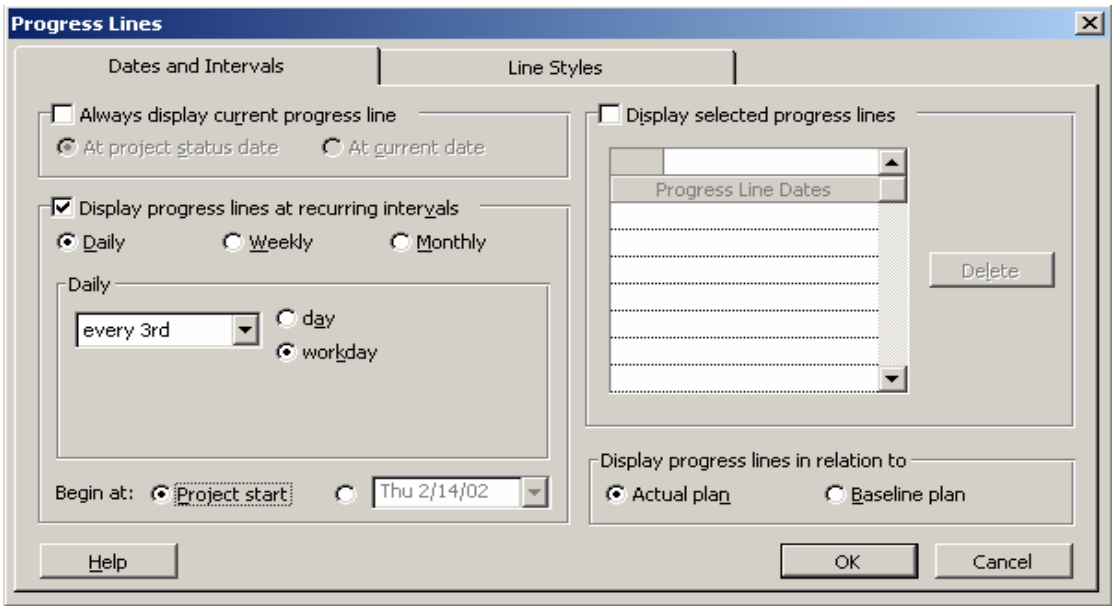


Figure 11.15: *Progress Lines in the Gantt Chart View*

Project Baselines

Project baselines are another way of depicting the progress of a project. Baselines are indicators that help you compare the actual start date with the planned start date of a project. They also help compare the actual cost incurred with the planned cost for a project. After preparing a project plan for tasks, you can save the plan as a baseline. This serves as an ideal plan for you to compare the actual performance of the project with the baseline that you save.

To save a project plan as a baseline, perform the following steps:

1. Click the **Tools** menu, and point to the **Tracking** command.
2. Select the **Save Baseline** command on the **Tracking** submenu.
3. In the **Save Baseline** dialog box, select the **Save baseline** and **Entire project** option buttons to save a baseline of the entire project.
4. Click **OK** to close the **Save Baseline** dialog box. You can also save a part of the project plan by saving a few tasks as baselines by clicking the Selected tasks option button.
   To view the deviation from a baseline performance to the actual performance of a project, perform the following step:

5. In the **Project Information** dialog box, click the **Statistics** button to view the project statistics for the Sample-Project.mpp dialog box. This dialog box is displayed in Fig 11.16.

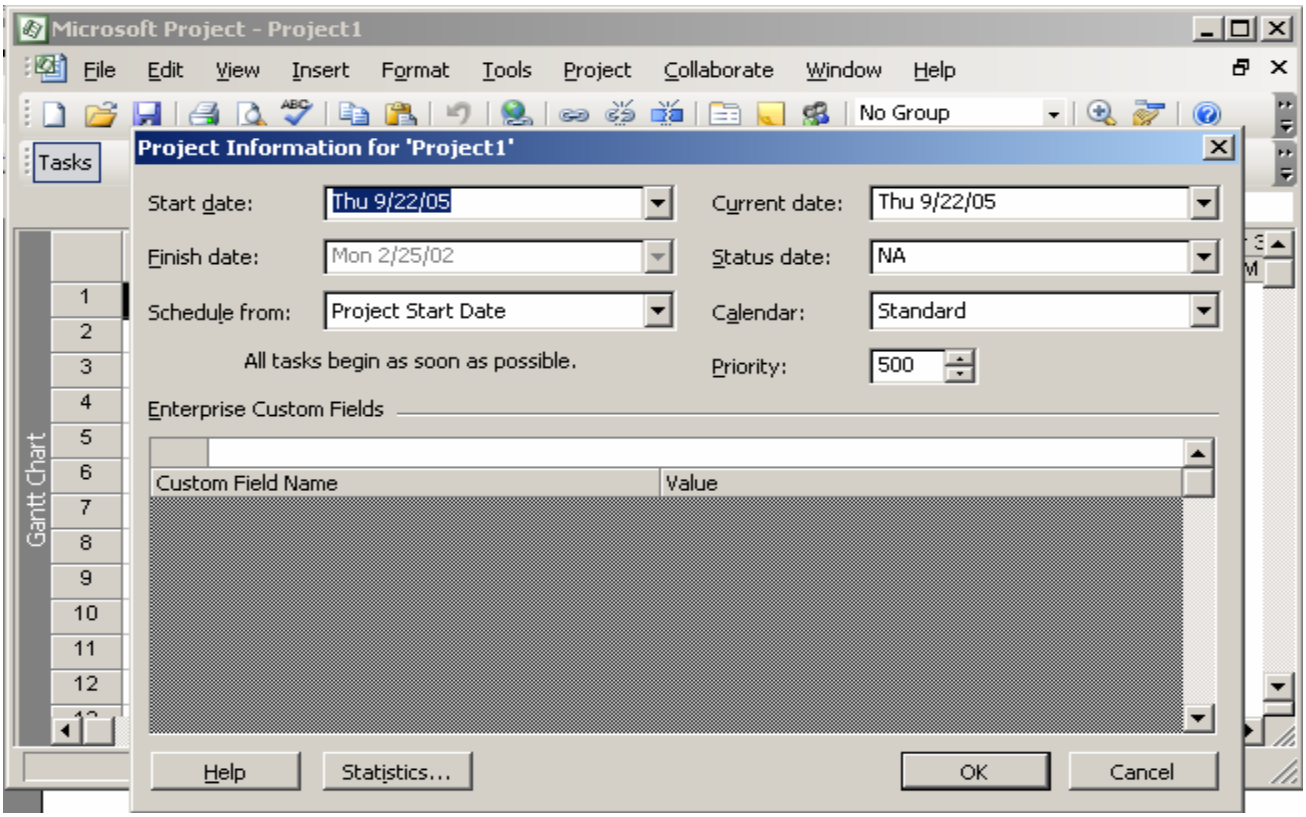


Figure 11.16: *Displaying Project Baseline Information*

The dialog box displays critical and summarized information about the variance values of the start and finish dates of a project, work completed in hours, cost, and duration in days. Higher the variance of a column, higher is the degree to which a project does not conform to the baseline plan. The actual values are displayed in the dialog box.

You can compare the baseline values for all these factors with the actual values of a task to conduct project plan analysis.

11.12 Recording and Updating Tasks

One way of tracking a project schedule is by determining the actual work done by a resource. After doing that, you compare it against the total work to be done initially. Using these factors, you can calculate the remaining work that needs to be done in the rest of the project cycle.

For example, you need to check the status of the Test component modules task in the XYZ Inc. project .This task exists under the Testing summary task. You use the Task Sheet view to record the actual work done by a resource and calculate the work that remains to be done. To do that, perform the following steps:

1. Select the task Unit testing for which you want to view the task completion status. This task exists under the Testing summary task.

2. Click the View menu.
3. Click the Table Entry command.
4. Click the Work command on the Table Entry submenu

After doing this, four fields appear: Work, Baseline, Actual, and Remaining.

You update the Actual field value with the current amount of work done. Microsoft Project automatically updates the Remaining value after comparing the Actual value with the value in the Work field. Alternatively, you can update the value of the remaining field to a number that you feel is correct. Consequently, Microsoft Project updates the Tota1figure in the Work field. It also displays the difference in the previous Work and the current Work, values in the Variance field.

For example, the Work field in Figure 17 displays the value 20. You update the Actual field with the value 13. Consequently, the remaining field would automatically show 7. However, you can modify the Remaining field to display the value 10 if seven days are not adequate. Notice that the Work field displays the value 27, and the value in the Variance field is 3.

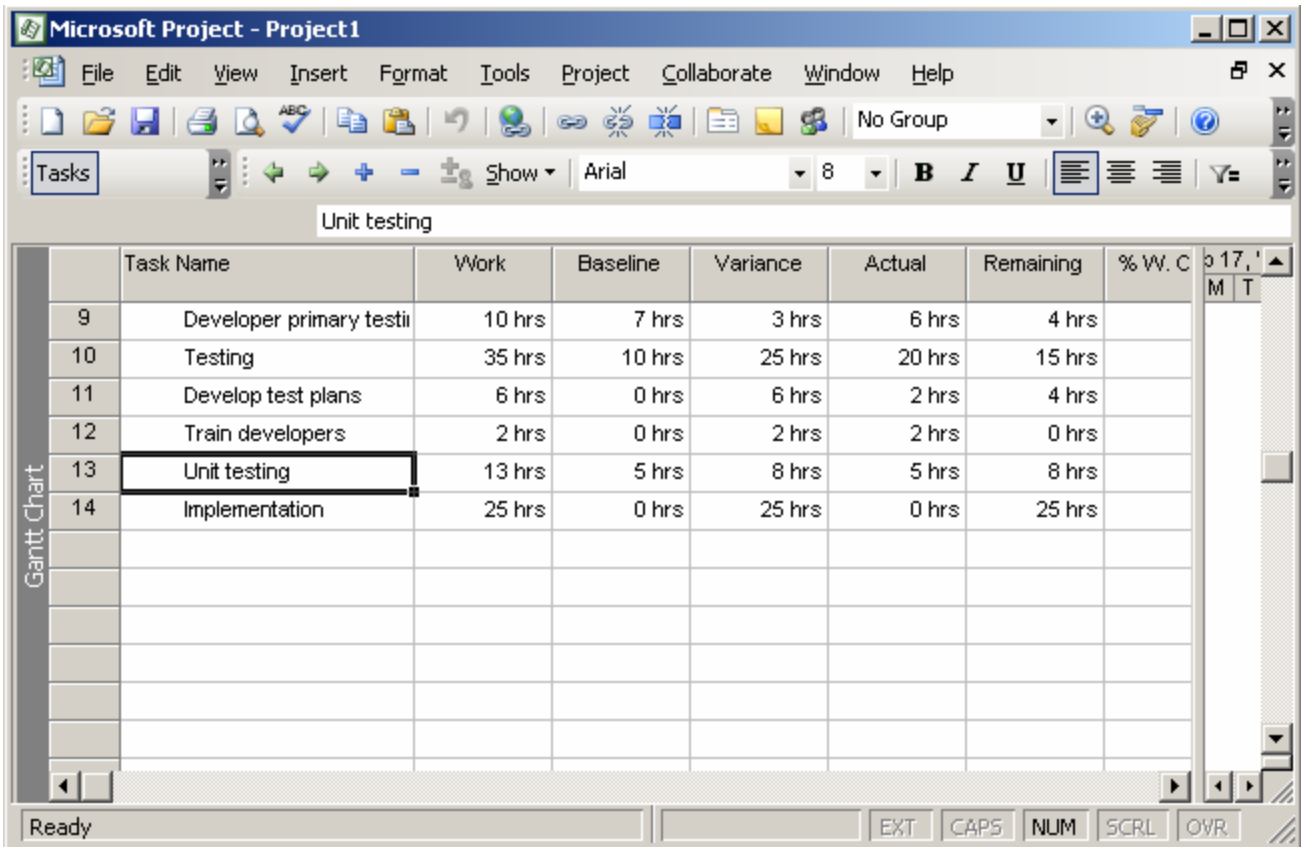


Figure 11.17: *Updating the Status of Tasks*

Note:
You can also update the percentage of work completed by specifying the value in the Percent complete spin box. The Percent complete spin box is present in the General tab of the Task Information dialog box. Consequently, Microsoft Project updates the actual and remaining work in the work sheet of tasks.

11.13 Rescheduling Tasks

Using Microsoft Project, you can reschedule the tasks of a project. Rescheduling involves modifying the start or finish dates of a task. For example, you need to reschedule the start date of the Create user manuals task. This task exists below the Development summary task. To reschedule an incomplete task, perform the following steps:

1. Create a task named Create user manuals under the Development task.
2. Make the Development task a summary task by using the left indent Icon.
3. Select the Create user manuals task.
4. Click the Tracking command on the Tools menu.
5. Click the Update Project command on the Tracking submenu, to display the Update Project dialog box. Figure 11.18 displays this dialog box.

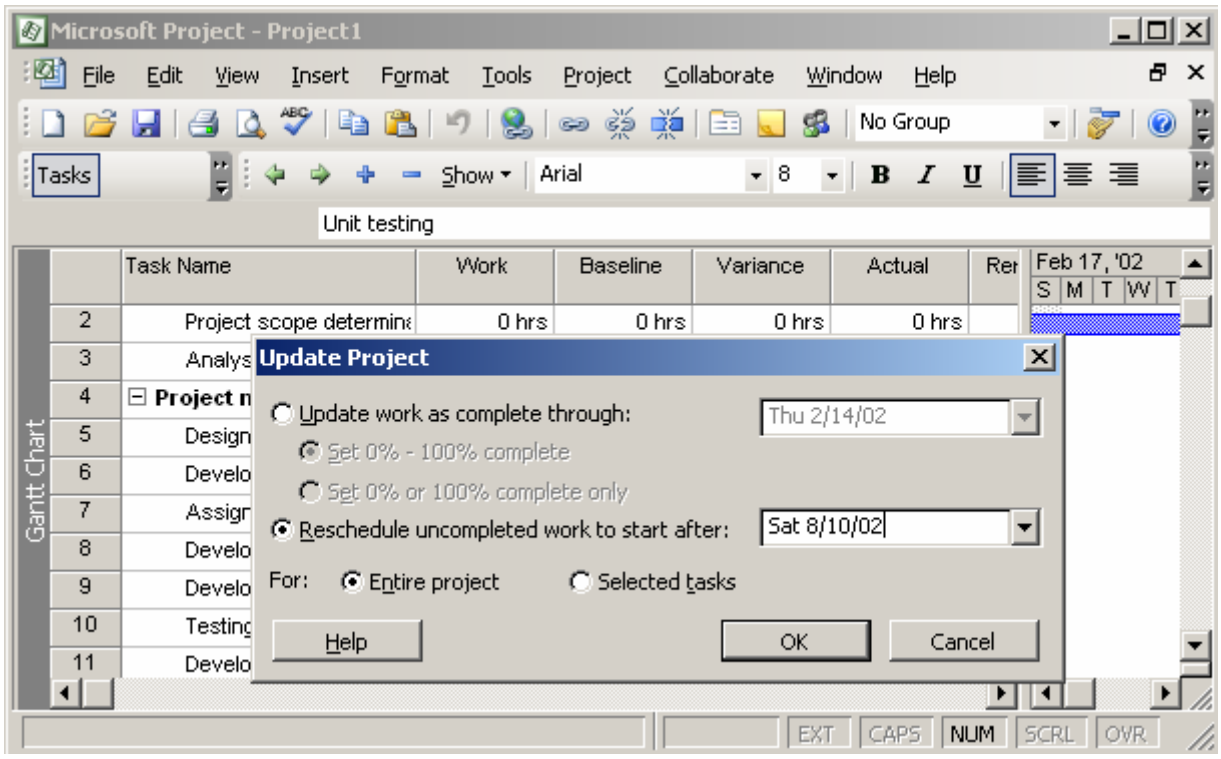


Figure 11.18: *Rescheduling Tasks*

6. In the Update Project dialog box, select the Reschedule uncompleted work to start after option. This option is used to reschedule an incomplete task.
7. Specify a new start date for the Create user manuals task.
8. Select the Selected tasks option to reschedule a task. If you need to reschedule entire project, choose the Entire project option.
9. Click OK.

11.14 Modifying the Duration of a Project Plan

During the course of a project, schedule, cost, and effort slippages are quite common. Therefore, you sometimes need to modify the project start and end dates. Normally the start date is modified for short cycle projects that need an early head start. Finish date is modified when you are unable to meet the project end date due to slippages. Assume that the project starts on February 15 and you change its finish date from June 10 to June 15. To do this, perform the following steps:

1. Click the **Project** menu
2. On the Project menu, click the **Project Information** command. This displays the Project Information dialog box.
3. In the Project Information dialog box, click the **Schedule from** list box.
4. Select the **Project Finish Date** option from the Schedule from list.

5. Click the **Finish date** box, and select June 15 from the calendar of the month June. This step is shown in Figure 11.19.
6. Click **OK**.

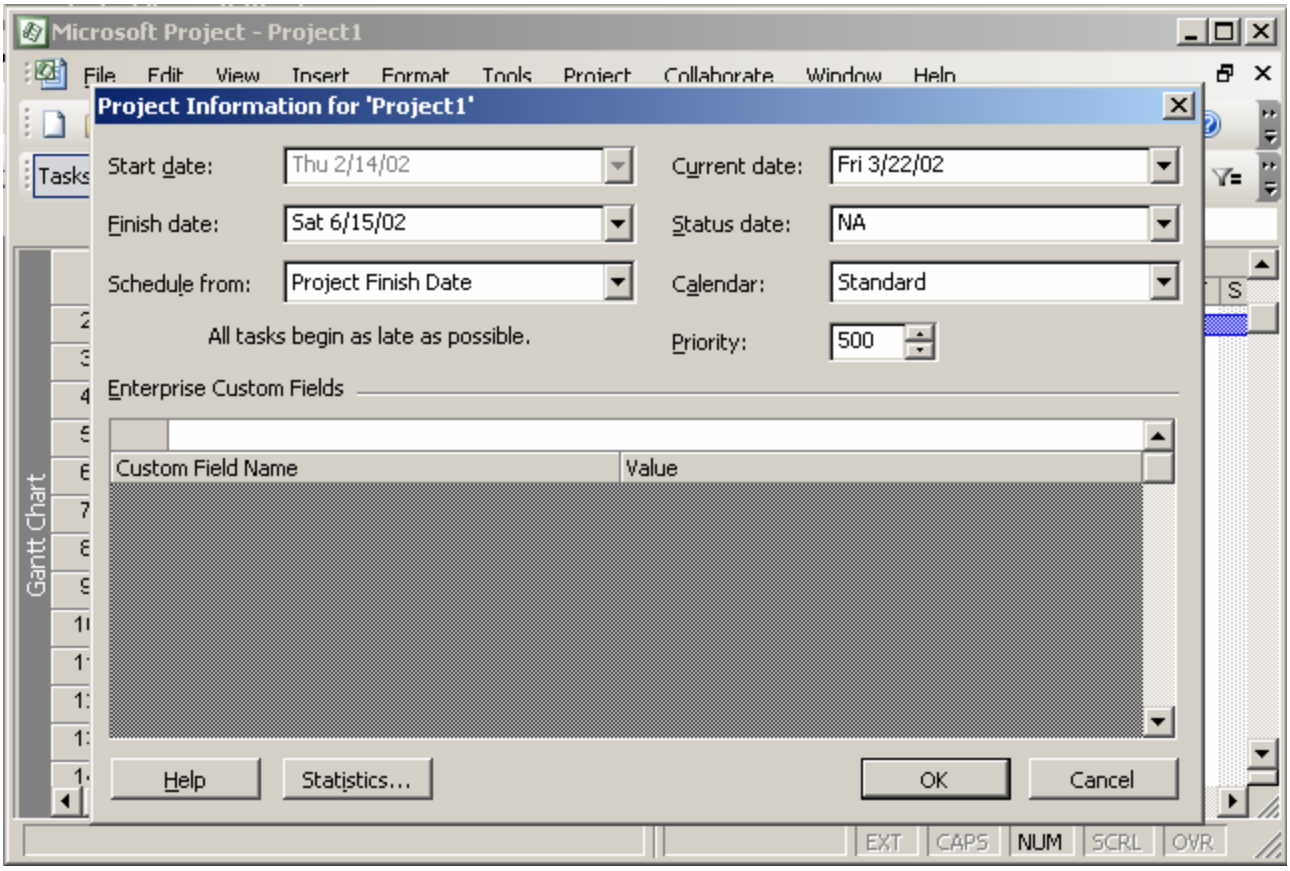


Figure 11.19: *Modifying the Plan Duration*

You have modified the project finish date from June 10 to June 15. You can view the change in the Gantt chart view of the project.

11.15 Displaying Project Information Using Reports

You create reports to display project-related information in a tabular format, which is easily comprehensible. Project reports provide a picture of the status of a project. The status of a project helps measure the progress of it. Reports can be prepared for a specific phase or for all the phases at one time.

Creating Standard Reports

Using Microsoft Project 2000, you can create two types of standard reports:

- Standard resource report: You create standard resource reports to display resource allocation and the costs incurred on them

- Standard task report: You create standard task reports to display information regarding tasks and the duration to complete each task in a project.

To create a report on the resources listed in the Sample-Project.mpp file, perform the fo1lowing steps:

5. Display the Resource Sheet view of Microsoft Project
6. Click the View menu.
7. Click the Reports command.
   This command displays the Reports dialog box. This contains the various report formats.
8. Select the Custom icon to create a new resource
   Figure 11.20 displays the Custom icon.

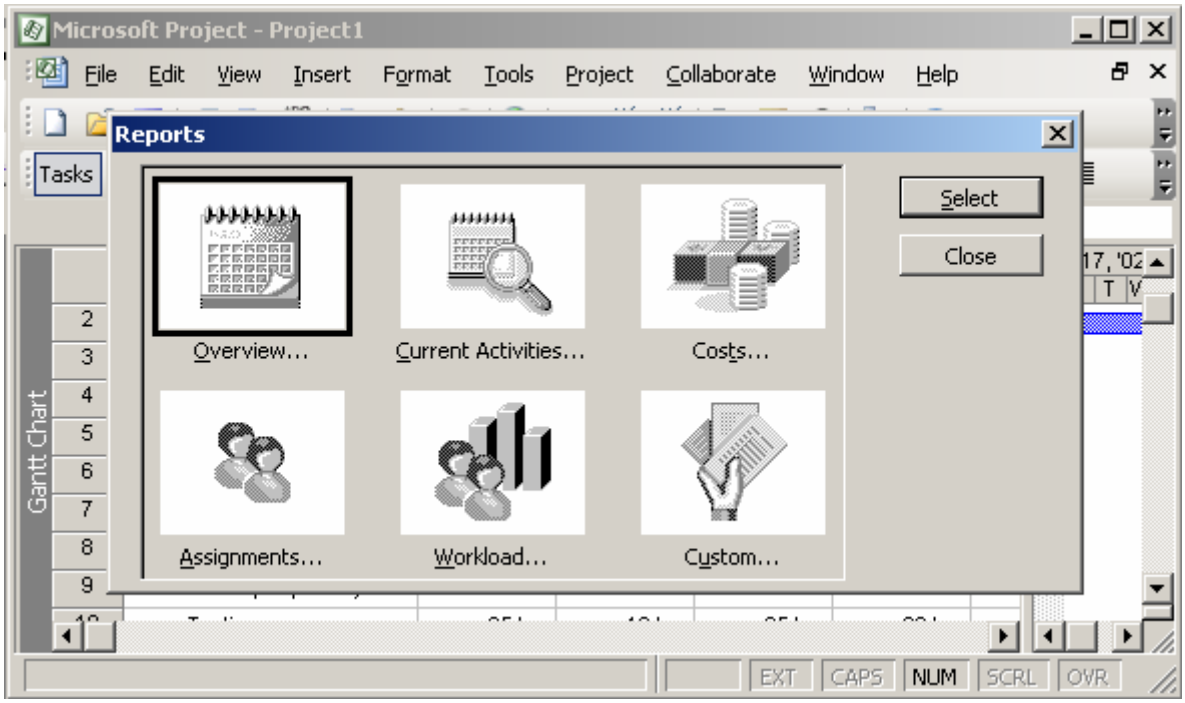


Figure 11.20: *Displaying the Custom Icon in the Reports Dialog Box*

5. Click the Select button.
   This displays the Custom Report dialog box.
6. Click the New button.

This way, you create a new type of a resource report that is not part of the list of reports displayed. The Define New Report dialog box is displayed as shown in Fig 11.21.

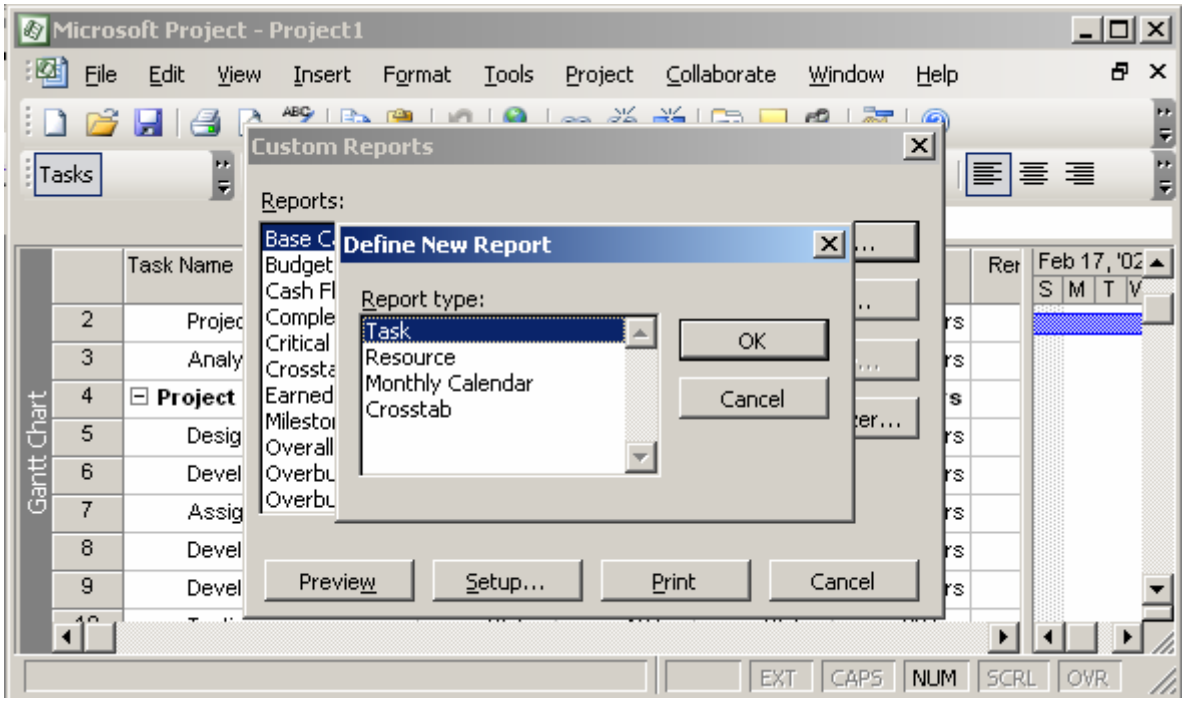


Figure119.21: *Defining New Reports*

7. In the Report type box, select the Resource option.
   You do this to create a report displaying the various resources used in a project.
8. Click OK to confirm your selection.
   The Resource Report dialog box is displayed as shown in Figure 11.22. In this dialog box, you specify the name of the report. In the same dialog box, you can sort the information to be displayed in the report.

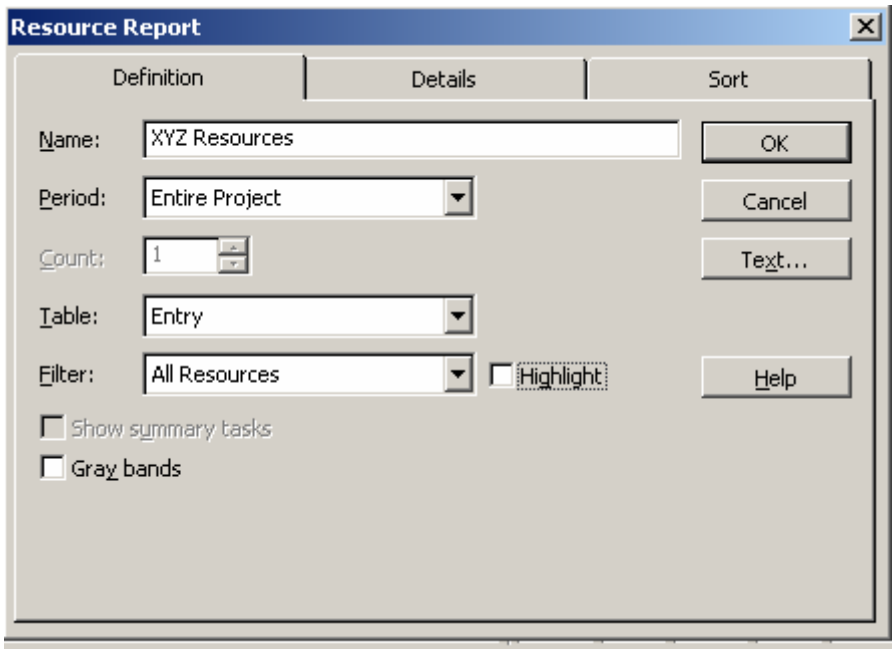


Figure 11.22: *Resource Report Dialog Box*

9. Click **OK** to confirm your specifications.
   After the Resource Report dialog box is closed, the Custom Reports dialog box is displayed.
10. Click the **Preview** button to view the report.
    The report is shown in Figure 11.23. To be able to view the report, you configure a printer to the computer that has Microsoft Project.

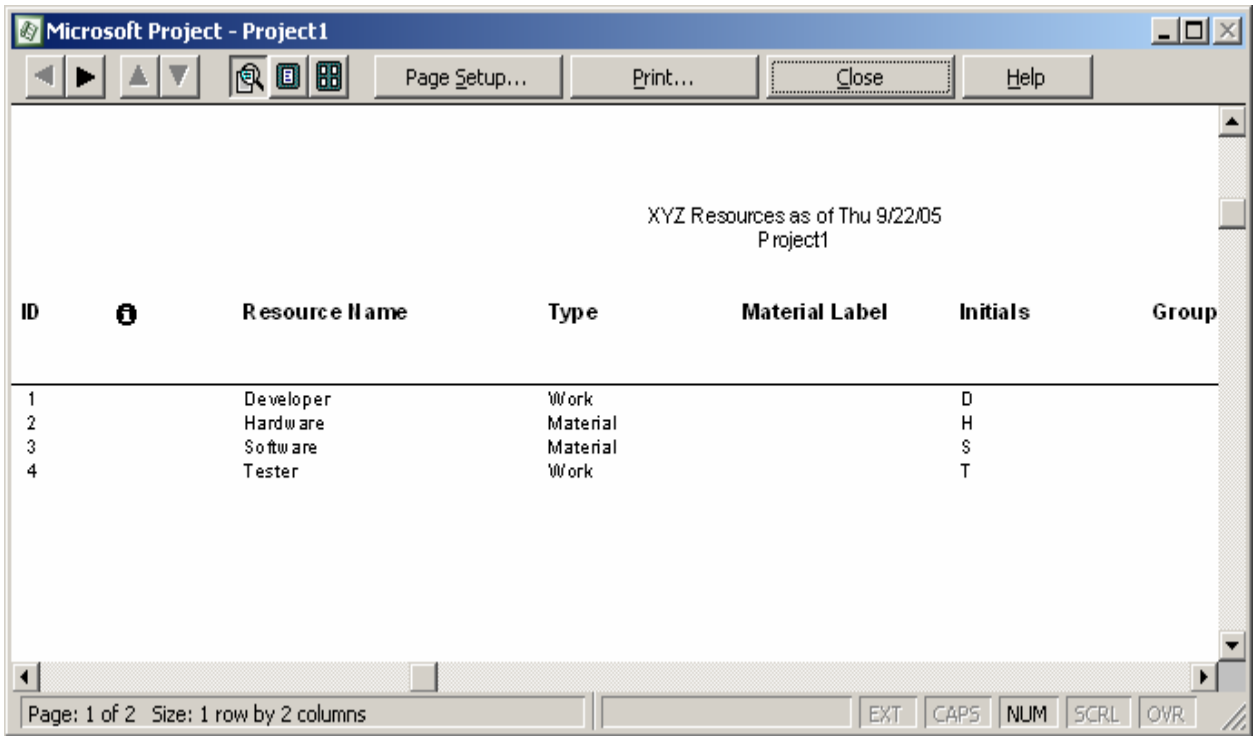


Figure 11.23: *Report on the Resources Listed in the Project1.mpp File*

Similarly, you can also view a report on tasks. To view the report on tasks, repeat the procedure *for* creating a report on resources in the Gantt chart view of Microsoft Project.

Modifying and Printing a Standard Report

You might need to modify a report to display more information or to display the existing information in a different format. To modify the ways in which a task is displayed, you Use the Task report dialog box.

You can modify an existing standard task or a resource report by clicking the Edit button on the Custom Reports dialog box. This displays the relevant reports dialog box.

For example, to modify an existing report to display only summary tasks and add gridlines to each task for the XYZ Inc. project, perform the following steps:

1. Click the Definition tab in the Task Report dialog box. The Task Report dialog box is displayed in Figure 11.24.

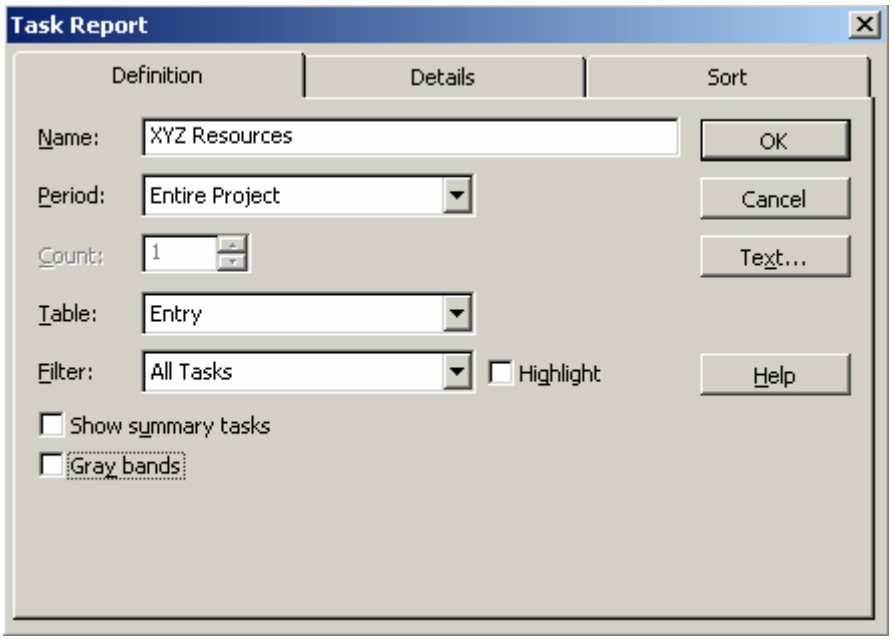


Figure11.24: *Modifying Reports*

2. Click the Show summary tasks check box.

   This check box is used to display only the summary tasks in a project. For example, in the current project file, selecting this check box would display only the broad-level tasks, Analysis, Design, Development, Testing, and Implementation and not the subtasks below each one of them.

3. Click the Details tab.
4. On the Details page, select the Gridlines between details check box.
   The Gridlines between details check box is shown selected in Figure 25.

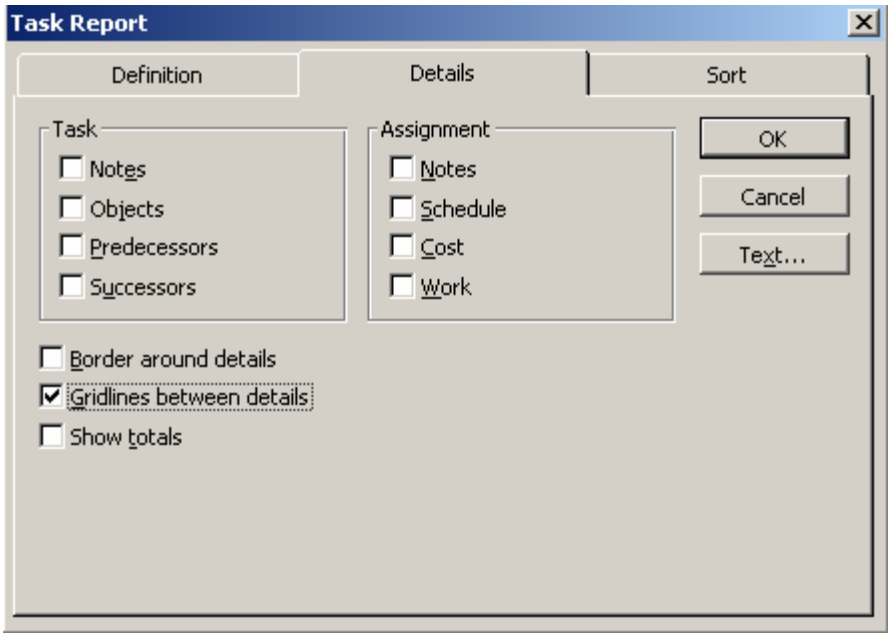


Figure 9.25: *Selecting Gridlines between details Check Box.*

5. Click OK to close the Task Report dialog box.

In Task Report dialog box, you can display specific fields in the Definition tab. In addition, you call also use the Details tab to display a summary of tasks in gridlines and resources.

You can also sort a report on different fields displayed on a report.

Printing a Standard Report

Printing a standard report is really simple. In the Define New Report dialog box, you click the Print button to print a report.

Figure 1126 displays the Define New Report dialog box. Clicking the Print button in the Custom Reports dialog box, displays the Print dialog box.

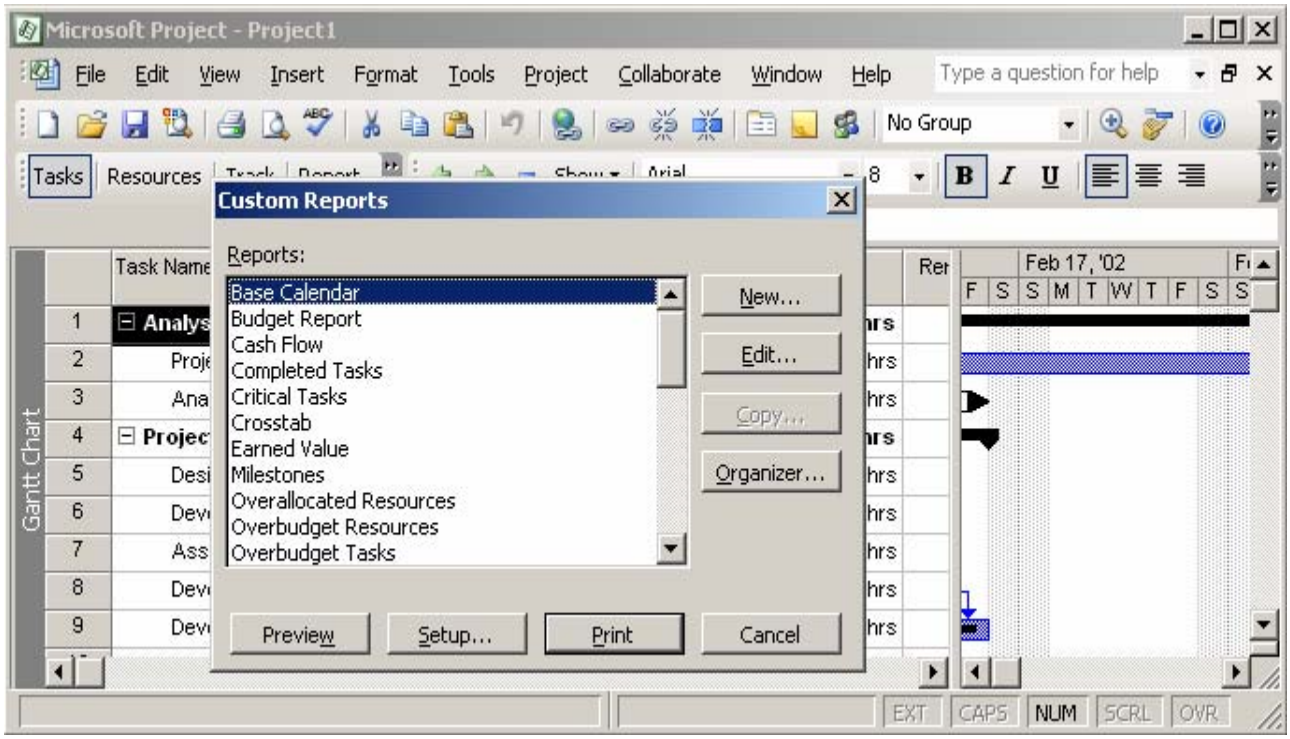


Figure 11.26: *Printing a Report*

Finally, you can see the Sample-Project.mpp file that you are creating for XYZ Inc. complete with all the components that you have added.

12 Commissioning & Migration

12.1 Software Implementation

After the software product is developed, the challenge lies in seeing the system work successfully at the user-end. Product implementation is the crucial activity of actually handing over the product to the users. The product moves from the theoretical environment of the development team to the practical environment of the user. Implementation begins with the preparatory activities of planning, training the user staff, and testing the system. After the preparatory activities are complete, the new system is installed and configured at the user end. Implementation marks the end of product development. After this, the product is handed over to the maintenance team.

Implementation Prerequisites
The purpose of implementation is to organize and implement the new or updated system. Most of the time, implementation is a contract-based activity. The contract provides details of the implementation activity. Implementation procedure includes details such as the scope, method, and duration of implementation. There are some prerequisites of implementation. They are finalizing the number of licenses, procedures for error logging, scheduling system changeover, and procedure for maintaining data backups. However, the primary task is to organize a team to monitor implementation.

An implementation coordination committee is set up to monitor implementation committee. Other committee members include line managers of other departments and members of the development team.

IThe committee performs various activities. It identifies various requirements of implementation process, such as arranging for software licenses, project software, and other standby facilities. The committee looks into the implications of the system environment, staff allocation for implementation tasks, and internal communication. The committee also devises alternative methods facilitating system changeover.

Key Implementation Plans
The various activities performed during implementation are preparing an implementation plan, a raining plan, an acceptance plan, and a system support plan.

a. Implementation Plan

The first task of implementation is the creation of the implementation plan. The implementation plan contains details of the methods of system transfer and the timescale for implementation. Details of hardware and software needed for implementation are also mentioned in the plan. The plan contains details of date, location, and timescale for implementation. The plan also mentions the prerequisites of implementation, such as the existing network environment and the availability of related software. You can look at an implementation plan in Table 12.1.

Table 12.1: *Implementation Plan*

Implementation Plan					
1) Implementation Schedule					
Serial No.	Location	Start Date	End Date	Days	Resources
2) Resource List					
Serial No.	Resource Name	Contact Name	Pager or Mobile No.	E-mail ID	Key Resource (Y/N)
3) Software Components for Installation					
3.1) Server					
Serial No.	Software Name	Version	Space Required	Location	
3.2) Client					
Serial No.	Software Name	Version	Space Required	Location	
4) Data Migration					
Serial No.	Data to be Migrated	Location of Server	Data Migration	Mode of Migration	Batch or Script

In the implementation plan, Implementation Schedule section records .details such as location and duration of implementation. The Resource List section has a list of the resources and their contact numbers. The plan has details of the server and clients in the Components of Installation section. Finally, the plan also contains data migration details.

b. Training Plan

Preparing a training plan is one of the tasks of implementation. The training plan contains details of the participants to be trained. It also mentions the focus area arid .the audience for die training session. Training schedule containing the training venue, training name, start date and end date is a part of the training plan. The plan conveys budgetary details and the roles and responsibilities of participants as well as the faculty. A training plan is displayed in Table 12.2. The plan has details of the focus areas of training, courses identified for training, and training schedules. Roles and responsibilities of trainer are also included in the training plan.

Table 12.2: *Training Plan*

Training Plan						
1) Focus Areas for Training						
Serial No.	Focus Area					
2) Training Courses						
Serial No.	Course Title	Duration	Category	Faculty	Details	Number of participants
3) Training Schedule						
Serial No.	Course Title	Code	Start Date	End Date	Venue	
4) Roles and Responsibilities						
Member	Role	Back-up	Responsibilities			

c. Acceptance Plan

The purpose of an acceptance plan is to determine the details of acceptance testing. The plan contains details such as the order of testing of functions, location, date, and duration of testing. Details of participating members, coordinators, and users are also mentioned in the plan. The acceptance plan lists the procedure for bug fixing, version control, and verification and validation details. Procedure for the creation of acceptance test cases and setting up software and hardware environment is also mentioned in the plan.

The acceptance plan in Table 12.3 contains details of the acceptance schedule, acceptance criteria, list of resources, and a bug report form.

Table12. 3: *Acceptance Plan*

Acceptance Plan							
1) Schedule							
Serial No.	Module	From Date	To Date	Days	Resource Name	Documentation	
2) Acceptance Criteria							
Serial No.	Item Name	Acceptance Criterion					
3) Resource List							
Serial No.	Resource Name	E-mail	Telephone/Pager	Key Resource (Y/N)			
4) Bug Reporting Procedure							
Serial No.	Module	Bug	Reported by	Report Date	Closure Date	Assigned To	Fixed On

d. System Support Plan

The implementation team prepares the system support plan either during or before product installation. The system support plan contains details of how the system will be supported after it is implemented. Typically, the support plan has a warranty clause. The warranty clause usually lasts for a year. The warranty covers details of people in the support activities, details of agreement, the response time, and support details.

Implementation Activities

After the prerequisites are in place, implementation activities begin in right earnest. Now, all the plans created for implementation are put into practice. Implementation activities include acceptance testing, user and operator training,

data migration, installation and configuration, product sign-off, product wind-up, post mortem, and product delivery.

i. User Acceptance Testing

You must have heard the proverb, "the proof of the pudding lies in the eating. This applies for software product also. The success of the software product depends on how the user responds to the product. One of the important implementation activities is user acceptance. This aims at ensuring that the system is working according to the user requirements. UAT is done to ensure that the desired product design matches the actual product design.

UAT is the final testing performed by the end user after using the product over a limited period me. As part of the implementation process, the project manager prepares the User Acceptance Test plan. The UAT plan is the last stage of the testing process where the user tests the product. The user tests the product and identifies whether it is acceptable or not.

Ideally, UAT is conducted in the same environment as the one in which the product is designed. UAT plan consists of the activities to be conducted, functions to be tested, the time, and environment in which software will be tested. Often test cases are attached with the plan. The user validates the software based on the test cases.

The executable product is installed in the test environment and configured for use. Then the user executes the test supplied. r the user tests the software product, the implementation team takes care of the bugs or items identified by the user. The bug fixing and closing of issues continue iteratively until all problems are solved. The software product is then ready for use. The user representative uses a signoff on the prescribed format to indicate satisfaction with the product. This marks end of the development activity. The final bug-free version is delivered to the user.

ii. Training

To familiarize the users with the new system, you need to train them. Apart from the users, there are the operators who also need to be trained to handle the system. Training for both the users and operators has separate goals. Training sessions aim at providing the participants with specific skills required in handling the new system. Training aids used are documents, icons, on1ine helps, and demonstrations or classroom sessions. The user training focuses on major m functions and the users' need to gain access to them. Some of the activities that a user might need to perform are simulating activities, analyzing data, drawing graphs, and searching for information. The user is trained on navigation skills, information searching skills, and security functions. The users are trained on the differences between the old and new activities to familiarize them with the differences.

The operator training focuses on the support functions of the system. The support functions include granting user access, granting file access, recovering damaged files, installing new devices and software, and making backups. Therefore, the operators' training focuses on two function areas. First, the operators learn to run the system, and second they learn to support the users.

Another important approach to training is documentation. Documentation can be in the form of user manuals and guides. The use of documentation in training will depend on the target audience. Therefore, you need to ensure that your development team does not neglect training and documentation. This should be planned and tracked from the project start as soon as the requirements analysis is complete. You also need to ensure that training and documentation for users and operators is updated periodically.

iii. Data Migration

At times, the user might want to transfer data from the old system to the new system during implementation. The implementation team migrates data from the old system to the new system. This task can be automated by using scripts and tools. Data migration is an important activity. As a project manager, you need to ensure that all data is successfully migrated. At times, data migration involves a large amount of data from heterogeneous sources. This entails lot of work, which might be beyond the handling capacity of the implementation team. Typically, in such situations, data migration activity is subcontracted to an external agency. If the data is in the form of documents, data entry operators could also be hired. Although the activity is subcontracted, as a project manager, you need to plan and monitor its progress continuously.

iv. Installation and Configuration

To make the system available to end users, the implementation team needs to install and configure the system. This involves making the system run and prepare it for actual work. The implementation team does this by deploying the system on user machines. During installation, the files, components, and executables are copied to the user machine.

Configuration is the act of making necessary changes in the system parameters that are specific for target users. This activity is performed along with installation. Installation and configuration activity can be automated by using many tools. You can perform installation and configuration at multiple locations by using tools. These tools create installable disks that can be used to install and configure. This enhances the productivity of the implementation team to a great extent.

At times, the project manager might need to outsource the installation activity. Then great care needs to be taken to ensure that correct versions are installed. At times, Web-based automatic installation facilities are used to simplify the process further. At times, the implementation team might need to install the

system at multiple locations simultaneously. The team prepares separate plans for each location. The implementation team also configures the installed system.

v. Product Sign-Off

At the end of acceptance testing, the user-representative gives the sign-off in the form of an acceptance letter. This marks the formal end of the implementation phase. This means that all issues raised during the acceptance testing are solved and closed. As a project manager, you need to ensure that the contractual obligations with the user are fulfilled. You need to verify that the client accepts all documents and deliverables as mentioned in the contract. You also ensure that the sign-off is according to defined guidelines and is in the form of an acceptance letter.

vi. Project Wind-Up

Project wind-up marks the formal closure of the development phase. This activity begins after obtaining a sign-off from the user. The key activities performed during project wind-up are analyzing what went wrong and what could have been better. Major learning from the project is summarized at the end of the project. Certain completion criteria are formalized at the project start. These criteria prevent the project from dragging on endlessly. Certain activities are performed to ensure that the project ends on time.

- Monitoring deliverables
- Monitoring progress
- Controlling change requests
- Monitoring work yet to be done

Usually a senior project manager supervises the project-wind up activities. During project wind-up, you need to gather feedback both internally as well as from the client. The various wind-up activities performed during project end include obtaining client feedback of the project. You also need to update and close the project plan. Another activity is updating project-related statistics at the end for resource estimation. You can determine the level of success of the project based on the quality of the delivered system and the level of satisfaction of all concerned.

During project wind up, the project manager also plans for the allocation of the project staff. The deliverables list is checked to ensure that all requirements are met. The project manager also verifies the financial completion of the project. A contract is also drawn to deal with the problems later for a defined period under a warranty. The project is evaluated for adherence to schedule, budget, technical goals, and client and project team satisfaction.

Another important project-end activity is data collection. Data accumulated during the project is collected and organized after completion. This data helps in resource estimation and quality assurance of future projects. The data collected regarding the actual cost per task, cost overrun and under-run per task, unplanned tasks, and planned task that were not required are estimated. Data about software change requests and bug statistics is also noted.

The performance of the project team is also evaluated at the end of the project. The team members are evaluated in terms of their ability to accomplish project tasks. The time taken, the quality achieved, and the adherence to cost and schedule is also noted. The team members are also evaluated based on their innovative approach to accomplishing a task. The ability of the team member to work as a team, communicate, and support each other is also judged. The team members are also asked to evaluate themselves in terms of their strengths and weaknesses.

vii. Post Mortem

The post mortem activities involve collecting valuable information at the end of the project. The knowledge gained from the project is documented for use by other projects. Similarly, the recommendations for enhancements are also documented. You also need to submit project- related items, hard copies, and files to the project library. Usually, post mortem activities are conducted among the team members. The team members exchange feedback about each other regarding interpersonal communication and work processes within the team.

Finally, the project manager prepares a project closedown report. This report contains suggestions on what should be done to improve software quality in the next project. This report contains details of what the team has failed to deliver and the lessons learned from the project. This report will help detect and eliminate long-term problems. The project closedown report contains the following details:

- Description of the project
- Success assessment
- Schedule, budget, and technical details
- Problems while implementing solutions
- Data updates regarding cost estimation

viii. Product Delivery

Product delivery marks the last leg of implementation. In this phase, the packaged software, documentation, and other items mentioned in the contract are passed on to the maintenance team. As a project manager, you need to ensure the smooth transfer of knowledge from the development team to the maintenance team. The maintenance activities are discussed in detail in the next section.

12.2 Software Maintenance

According to surveys, changeability is one of the most important attributes of complex, multifunction software systems. This calls for continuous adjustments of the software to suit the changing environment. While changes are inevitable, it is not always possible to create products foreseeing the changes. Therefore, the product needs to be changed according to the changing scenario. You might need to upgrade the software product based on new system requirements or remove redundant functionality. The need for a change brings software maintenance into the picture. Maintenance activities can either be in-house or outsourced.. In the case of in-house maintenance, the development team performs the maintenance activities. On the other hand, an independent maintenance team performs outsourced maintenance activities.

Software maintenance deals with the sustenance of a piece of software after it is released. Maintenance activity is needed when errors are detected after the software product is released. Changing requirements of the software product also require maintenance activity to cater to the changes.

Maintenance activity assumes greater importance for a project manager. This is because managing software maintenance is more challenging than managing software development. Consider an example. You are maintaining a banking operation. A small Program change is introduced as part of the maintenance activity. Suppose this change results in an incorrect interest calculation. You can well imagine the kind of confusion this error will create, not to mention the loss of client faith and goodwill in the bank. Therefore, as a project manager, you need to be extra cautious while managing a maintenance project.

Types of Maintenance Activities

Maintenance is an important phase in SDLC: The need for maintenance activities has increased with the increase in software packages. Maintenance activities include correction and prevention of defects, enhancements to incorporate changing needs, and porting of applications and adaptability. The changing hardware and software scenarios have increased the demand for software maintenance. Today, in the entire SDLC of a software application, the maintenance effort is about 80 percent and the development effort is only 20 percent.

There are four types of maintenance activities:

- Corrective
- Adaptive
- Perfective
- Preventive

Corrective maintenance is about fixing bugs. This takes approximately 17 percent of the maintenance time. During corrective maintenance, the existing code is used to correct the fault that causes the code to deviate from its documented requirements. Here, the focus is on fixing defects.

During adaptive maintenance, the existing code is changed to adapt the new features and functionality. These new features are usually part of a new release of a code. This change normally takes 18 percent of the maintenance time.
Perfective maintenance improves the maintainability of the code. During this activity, the code is restructured to make it easily understood and to remove ambiguities. The enhancement of code occupies 60 percent of the maintenance time.

Preventive maintenance is undertaken to protect the code against failure. Here, the focus is on adhering to coding standards and reduce the chances of code failure. Preventive maintenance .activity takes around 5 percent of the maintenance time.

In the above list of maintenance activities, you can see that out of total maintenance effort, only 20 percent is spent on corrective maintenance and 80 percent on the rest. This refutes the popular belief that maintenance activities is all about fixing mistakes.

Maintenance Activities
To perform maintenance activities, the maintenance team needs to acquire the business and technical knowledge of the client systems. The maintenance process is usually divided into three phases.

- Initiation Phase
- Preparation Phase
- Execution Phase

Initiation Phase
The initiation phase starts with the commencement of maintenance activities. It is primarily a knowledge acquisition phase. During this phase, the maintenance team takes over the system from the development team. The maintenance team familiarizes itself with the ways and functioning of the system. This is essentially a phase of knowledge transfer from the development team to the maintenance team. The major activities of the initiation phase are baseline assessment and operating procedures. During baseline assessment, the interfacing and communication methods between the development and maintenance teams are defined. The user-coordinator from the user side and the information technology help from the maintenance team are identified. During this phase, the plan for acquiring the systems knowledge is also finalized.

During the operating procedures activities, the maintenance team understands the application and the functionality involved in the maintenance task. The

maintenance team reviews the current operating procedures of the system. The team also obtains details about the environment set up, deployment of software, the resource allocation process, and priority settings. Other' activities such as root cause analysis, work assignment to the development teams, and updating of documentation procedures is also done. The maintenance team refers to the problem management procedures, on-call or escalation procedures; and operational process for handling: faults. It also makes decisions based on the severity of the faults, application criticality, and infrastructure issues.

The maintenance team finalizes procedures for transferring support work from the development team. The maintenance team members are initiated to hands-on exposure of the systems. The maintenance team finalizes the quality systems and standards and guidelines to be followed.

Preparation Phase

The preparation phase begins after the initiation phase. During this phase, the maintenance team sets up the administrative and support procedures needed for maintenance activities. In this phase, the environment is set up and hardware, software, and network are made ready for operation. All the procedures decided and finalized during the initiation phase are put into practice in this phase. According to the documentation procedures finalized in the initiation phase, the latest documentation is kept in the library. During this phase, bug reporting, problem solving, and other support methodologies are established. The security measures are also put in place and practiced during this phase.

Execution Phase

The execution phase follows the preparation phase. In this phase, the maintenance team starts executing the maintenance activities after the preparation in the earlier phase. During this phase, the maintenance team is in close touch with the development team for a small period of time. Such communication is recommended to ensure better coordination.

Maintenance Curve

Maintenance activity begins after product implementation. The effort spent on maintenance varies over a period of time. Normally, maintenance effort is very high immediately after implementation. The system faces numerous changes during this time. The changes could be in the form of bugs, changes in network environment, or problems caused by user mishandling. During this time, the effort spent on maintenance activities is very high. With time, the system stabilizes and maintenance activities also slowdown. During this period, the users are more or less satisfied with the system. The system also runs smoothly without requiring any major changes. Therefore, maintenance activity is at its lowest during this phase.

However, over a period of time, business processes change and the system also depreciates. During this period, major changes start coming in. The changes are in the form of enhancements and additions of new systems or modules.
These changes take place continuously to keep the system up-to-date with the changing needs and requirements. Maintenance activities are also at a maximum during this phase.

However, soon the system becomes obsolete and is phased out. Along with the system phase- out, maintenance activities also come to an end.

Figure 12.1 shows the variation of maintenance activity with time. In the curve, maintenance effort is very high during the initial phase. Then the maintenance effort stabilize during the maturity phase. Finally, the maintenance effort increases sharply just before system phase out. The system is phased out to pave the way for new systems.

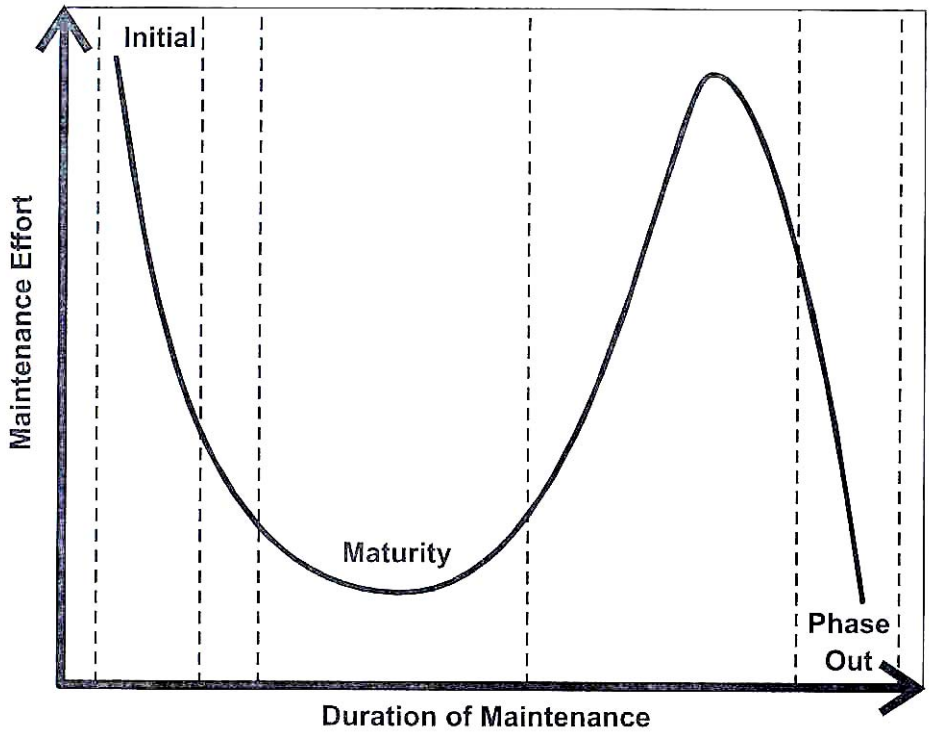


Figure 12.1: *Maintenance Curve*

Maintenance Process
There are certain prerequisites that you need to follow before beginning with the maintenance process. All documentation needs to be complete and up-to-date. All existing items need to be configured, and the latest copy of the code should be available. The hardware, software, and network environment for testing and maintenance should be made available. Before beginning with the maintenance process, the maintenance team is identified. The user representative from the user side is identified and the maintenance team is constituted. The maintenance team

consists of the head of IT, the project coordinator, and team members. The maintenance project 1anager is the single point contact between the user representative and the maintenance team. The project manager categorizes the type of maintenance activity to be performed based on specifications. The project manager ensures proper communication between all the entities 1volved in maintenance activity. To ensure that the maintenance process runs smoothly, there is a thorough checking of all deliverables. Maintenance activity is the most critical because the system is running and already in place. Therefore, the project manager needs to ensure that no error arises during the maintenance process.

The common approach followed in the maintenance process is as follows. After the user makes a request for change, the maintenance team performs an impact analysis. Then the team plans for the system release. The system goes through design changes, coding, and testing. Finally, the system is released. The maintenance activities undertaken are corrective maintenance, adaptive maintenance, perfective maintenance, and preventive maintenance.

Corrective Maintenance Process

The corrective maintenance process begins with the user representative reporting bugs to the project coordinator by using the problem report form. The project coordinator analyzes the problem and assigns the problem to one of the team members for corrective action. If the bug fixing leads to any changes in the system, the change is carried out based on the change request process. The team member requests the SCM team to check out the item to be modified. Then the bugs are fixed and tested in the test environment. After this, the user acceptance is carried out for the successful completion of the corrective maintenance process. Finally, the SCM team releases the modified item. All related documents are modified to reflect the changes and the project coordinator closes the problem report.

Adaptive Maintenance Process

In the adaptive maintenance process, changes are made to the system to accommodate changes in its external environment. Adaptive maintenance includes work related to enhancing software functionality. During this process, certain system changes, additions, insertions, deletions, and modifications are included to meet the evolving needs of the user and the environment. At times, the original environment of the software product might change. For example, there might be changes in operating system, business rules, and external product characteristics. Adaptive maintenance refers to modifications that adapt to these changes. The project coordinator plans the changes to be made. The detail plan and the quality goals are defined for the changes. The team is formed based on the quantum of work. The change request procedure is followed to carry out the changes. Finally, the changes are deployed based on a deployment strategy.

Perfective Maintenance Process

Perfective maintenance refers to enhancements that make the product better, faster, smaller, better documented, better structured with more functions and reports. It includes all efforts to improve the quality of software. Perfective maintenance extends the software beyond its original functional requirements. During perfective maintenance, the user or the system personnel initiates the changes to the system. The initiator raises the change request to the project coordinator. The changes could be a result of enhancements of new functionality in the existing system.

After tracking-the changes, an impact analysis is done and a detail plan is made to carry out the changes. Based on the amount *of* changes, a dedicated member or a team works on the changes. The testing is done in the testing environment- and changes are implemented in the system based on the deployment strategy.

Preventive Maintenance Process

Preventive maintenance activities make changes to software programs so that they can be more easily corrected, adapted, and enhanced. During the preventive maintenance process, the project coordinator identifies the potential risks. Recommendations are made to the management for approval of the maintenance actions. The project coordinator prepares the project detail plan for carrying out the changes. The preventive maintenance team is formed based on the quantum of work. The team implements the change based on the request procedure. Finally, deployment happens based on the deployment strategy.

The maintenance process begins when the user initiates a request for a change. The process ends when the system changes are tested and the change is released for operation. In between, there are many activities that are coordinated and planned using change management.

Reengineering

Reengineering is a process of abstraction of a new system from an old system. Reengineering is the modification and evolution of software product to meet the constantly changing business requirements. Failure to reengineer can prevent an organization from remaining competitive and persistent. This is because every system faces the risk of being phased out sooner than they were earlier. Therefore, you need to develop systems that support multiple platforms and are open to modifications and changes. Here, the role of reengineering comes into play. Reengineering comprises reverse engineering and forward engineering.

Reverse Engineering

Reverse engineering is a process of recovering the design of the running system from the source code. Reverse engineering involves analyzing a software system to achieve two objectives. First, reverse engineering helps to identify the system components and their Interactions. Second, it helps make representations of the system on a different, higher level of abstraction.

Suppose, you want to retrieve the design or a product whose design document is not available. You can recover the design by working backwards on the product from the source code to the design requirements. At times, you might need to reengineer programs developed many years back. The specifications of such programs are either unclear or was not developed at all. In such a situation, you use reverse engineering to analyze the program at a higher level of abstraction. Therefore, you perform reverse engineering to add a significant functionality to the old program. You can also use it to make the system cost-effective and to adopt new technologies.

Certain prerequisites are necessary for reverse engineering. The existing system should be in a running condition. The latest source code should be available. During reverse engineering, the reverse engineering team gathers information about the source code and the design and specifications documents. They also obtain information about the knowledge and experience of the developer. Then the team studies the existing code and the existing documents. Then they create a draft document from the existing code. The team discusses with the users and modifies the draft document based on the discussions.

To perform requirement reengineering, the team first prepares a prototype. After this, it provides the demo of the prototype to the users. Then the system requirement specification document is modified based on the discussions. The team obtains approval from the users on the requirements.

During design reengineering, the team studies the existing data model and the new requirements. Then they map the requirements with the existing data model and conduct gap analysis. After this, they modify the data model based on the gap analysis. The team studies the behavior of the existing programs and prepares program specifications based on the behavior and the existing code. Then the unwanted functionality is removed from the program specifications. The program is then modified based on the new requirements.

During construction reengineering, the team comments on the unwanted code from the programs. Then they add remarks in the filtered code, and modify the programs based on the new functionality. Finally, the team tests the reengineering process. Testing involves preparing the unit level test cases, the test code, and fixing the defects. Then the code for integration testing is released. The various testing methodologies followed are conducting integration testing, functional testing, independent testing, user acceptance testing, and releasing the code for deployment.

Reverse engineering helps understand the processing of the system, understand data, and eliminate dead code, bad coupling, and no documentation. It also reduces the maintenance efforts. The tools used for reverse engineering are Rigi and Refine Language Tools, static analyzer, and test editor.

Forward Engineering
Forward engineering is the process of building a new program from the requirements and design specifications of an old program. During forward engineering, old design information is recovered from existing software and is used to reconstitute the existing system. In the process, the quality of the existing system is enhanced and new functions are added. Therefore, forward engineering moves from a high-level abstraction and design to a low-level implementation. Forward engineering is used to improve the overall performance of the system.

Figure 12.2 displays the reengineering process.

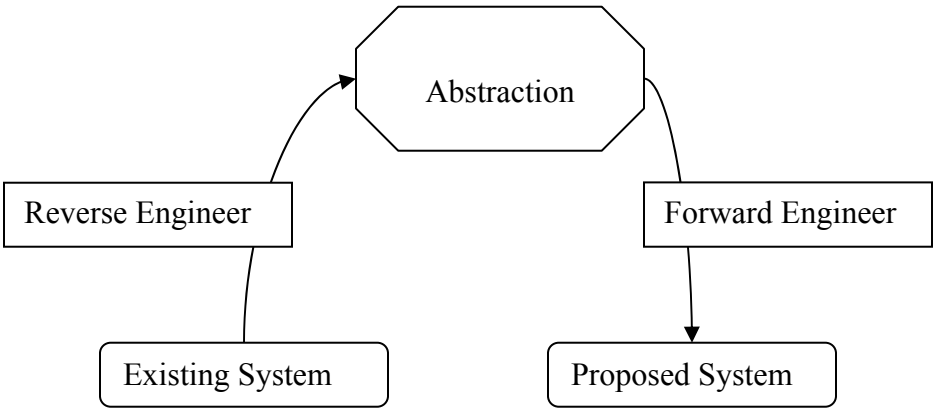


Figure 12.2: *Reengineering Process*

Maintenance: Best Practices
In addition to the maintenance process, there are some maintenance best practices. A maintenance process is successful if the testing and production environment are the same. Version control is strictly implemented using version control tools, such as VSS and PVCS. You need to check that the deployment strategy is clearly defined and tested and the documentation is kept up-to-date.

The members of the maintenance team management should be rotated in different business areas to enhance efficiency. It is very essential to identify backups of the key personnel. There should be a balance between the new development and problem fixing. Ensure proper training, and use the latest technology. Knowledge sharing sessions also help in improving communication within the team. There should be an online help facility for the maintenance team, and helpdesk tools should be available to track the problems. Periodic causal analysis of the nature of problems and corrective action need to be conducted. To ensure successful maintenance process, you need to conduct periodic quality reviews and monitor the quality goals of the team.