Module I

1

OVERVIEW OF ITS PROJECT MANAGEMENT

Unit Structure

1.0 OBJECTIVE

In this unit you are going to learn:

- IT Project Management
- Project Management Framework
- Role of IT Project Manager
- Project System View
- How to manage stakeholder?
- Project phases and lifecycle

Project management has been practiced since early civilization. Until the beginning of twentieth century civil engineering projects were actually treated as projects and were generally managed by creative architects and engineers. Project management as a discipline was not accepted. It was in the 1950s that organizations started to systematically apply project management tools and techniques to complex projects. As a discipline, Project Management developed from several fields of application including construction, engineering, and defense activity. Two forefathers of project management are commonly known: Henry Gantt, called the father of planning and control techniques who is famous for his use of the Gantt chart as a project management tool; and Henri Fayol for his creation of the five management functions which form the foundation of the body of knowledge associated with project and program management. The 1950s marked the beginning of the modern Project Management era. Project management became recognized as a distinct discipline arising from the management discipline.

1.1 WHAT IS A PROJECT?

All of us have been involved in projects, whether they be our personal projects or in business and industry. Examples of typical projects are for example:

- Personal projects:
➢ obtaining an MCA degree
➢ writing a report
➢ planning a party
➢ planting a garden
- Industrial projects:
➢ Construction of a building
➢ provide electricity to an industrial estate
➢ building a bridge
➢ designing a new airplane

Projects can be of any size and duration. They can be simple, like planning a party, or complex like launching a space shuttle.

1.1.1 Project Definition:

A project can be defined in many ways :

A **project** is “a temporary endeavor undertaken to create a unique product, service, or result.” Operations, on the other hand, is work done in organizations to sustain the business. Projects are different from operations in that they end when their objectives have been reached or the project has been terminated.

A project is *temporary*. A project’s duration might be just one week or it might go on for years, but every project has an end date. You might not know that end date when the project begins, but it’s there somewhere in the future. Projects are not the same as ongoing operations, although the two have a great deal in common.

A project is an *endeavor*. Resources, such as people and equipment, need to do work. The endeavor is undertaken by a team or an organization, and therefore projects have a sense of being intentional, planned events. Successful projects do not happen spontaneously; some amount of preparation and planning happens first.

Finally, every project creates a *unique product* or *service*. This is the **deliverable** for the project and the reason, why that project was undertaken.

1.2 PROJECT ATTRIBUTES

Projects come in all shapes and sizes. The following attributes help us to define a project further:

- *A project has a unique purpose.* Every project should have a well-defined objective. For example, many people hire firms to design and build a new house, but each house, like each person, is unique.
- *A project is temporary.* A project has a definite beginning and a definite end. For a home construction project, owners usually have a date in mind when they'd like to move into their new homes.
- *A project is developed using progressive elaboration or in an iterative fashion.*
- Projects are often defined broadly when they begin, and as time passes, the specific details of the project become clearer. For example, there are many decisions that must be made in planning and building a new house. It works best to draft preliminary plans for owners to approve before more detailed plans are developed.
- *A project requires resources, often from various areas.* Resources include people, hardware, software, or other assets. Many different types of people, skill sets, and resources are needed to build a home.
- *A project should have a primary customer or sponsor.* Most projects have many interested parties or stakeholders, but someone must take the primary role of sponsorship. The **project sponsor** usually provides the direction and funding for the project.
- *A project involves uncertainty.* Because every project is unique, it is sometimes difficult to define the project's objectives clearly, estimate exactly how long it will take to complete, or determine how much it will cost. External factors also cause uncertainty, such as a supplier going out of business or a project team member needing unplanned time off. This uncertainty is one of the main reasons project management is so challenging.

1.3 PROJECT CONSTRAINTS

Like any human undertaking, projects need to be performed and delivered under certain constraints. Traditionally, these constraints have been listed as scope, time, and cost. These are also referred to as the Project Management Triangle, where each side represents a constraint. One side of the triangle cannot be changed without impacting the others. A further refinement of the constraints separates product 'quality' or 'performance' from scope, and turns quality into a fourth constraint.

The time constraint refers to the amount of time available to complete a project. The cost constraint refers to the budgeted amount available for the project. The scope constraint refers to what must be done to produce the project's end result. These three constraints are often competing constraints: increased scope typically means increased time and increased cost, a tight time constraint could mean increased costs and reduced scope, and a tight budget could mean increased time and reduced scope.

The discipline of project management is about providing the tools and techniques that enable the project team (not just the project manager) to organize their work to meet these constraints.

Another approach to project management is to consider the three constraints as finance, time and human resources. If you need to finish a job in a shorter time, you can allocate more people at the problem, which in turn will raise the cost of the project, unless by doing this task quicker we will reduce costs elsewhere in the project by an equal amount.

1.3.1 Time:

For analytical purposes, the time required to produce a product or service is estimated using several techniques. One method is to identify tasks needed to produce the deliverables documented in a work breakdown structure or WBS. The work effort for each task is estimated and those estimates are rolled up into the final deliverable estimate.

The tasks are also prioritized, dependencies between tasks are identified, and this information is documented in a project schedule. The dependencies between the tasks can affect the length of the overall project (dependency constraint), as can the availability of resources (resource constraint). Time is not considered a cost nor a resource since the project manager cannot control the rate at which it is expended. This makes it different from all other resources and cost categories.

1.3.2 Cost:

Cost to develop a project depends on several variables including : labor rates, material rates, risk management, plant (buildings, machines, etc.), equipment, and profit. When hiring an independent consultant for a project, cost will typically be determined by the consultant's or firm's per diem rate multiplied by an estimated quantity for completion.

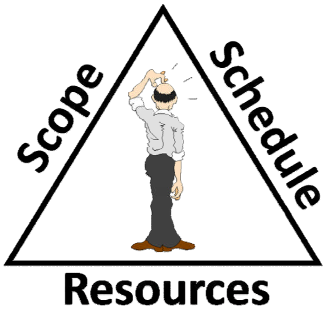


Figure 1.1 : The Project management Triangle

1.3.3 Scope:

Scope is requirement specified for the end result. The overall definition of what the project is supposed to accomplish, and a specific description of what the end result should be or accomplish can be said to be the scope of the project. A major component of scope is the quality of the final product. The amount of time put into individual tasks determines the overall quality of the project. Some tasks may require a given amount of time to complete adequately, but given more time could be completed exceptionally. Over the course of a large project, quality can have a significant impact on time and cost or vice versa.

Together, these three constraints viz. Scope, Schedule & Resources have given rise to the phrase "On Time, On Spec, On Budget". In this case, the term "scope" is substituted with "spec(ification)"

1.4 WHAT IS PROJECT MANAGEMENT

Project management is “the application of knowledge, skills, tools and techniques to project activities to meet the project requirements.” The effectiveness of project management is critical in assuring the success of any substantial activity. Areas of responsibility for the person handling the project include planning, control and implementation. A project should be initiated with a feasibility study, where a clear definition of the goals and ultimate benefits need to be determined. Senior managers' support for projects is important so as to ensure authority and direction throughout the project's progress and, also to ensure that the goals of the organization are effectively achieved in this process.

Knowledge, skills, goals and personalities are the factors that need to be considered within project management. The project manager and his/her team should collectively possess the necessary and requisite interpersonal and technical skills to facilitate control over the various activities within the project.

The stages of implementation must be articulated at the project planning phase. Disaggregating the stages at its early point assists in the successful development of the project by providing a number of milestones that need to be accomplished for completion. In addition to planning, the control of the evolving project is also a prerequisite for its success. Control requires adequate monitoring and feedback mechanisms by which senior management and project managers can compare progress against initial projections at each stage of the project. Monitoring and feedback also enables the project manager to anticipate problems and therefore take pre- emptive and corrective measures for the benefit of the project.

Projects normally involve the introduction of a new system of some kind and, in almost all cases, new methods and ways of doing things. This impacts the work of others: the "users". User interaction is an important factor in the success of projects and, indeed, the degree of user involvement can influence the extent of support for the project or its implementation plan. A project manager is the one who is responsible for establishing a communication in between the project team and the user. Thus one of the most essential quality of the project manager is that of being a good communicator, not just within the project team itself, but with the rest of the organization and outside world as well.

1.4.1 Features of projects:

- Projects are often carried out by a team of people who have been assembled for that specific purpose. The activities of this team may be co-ordinated by a project manager.
- Project teams may consist of people from different backgrounds and different parts of the organisation. In some cases project teams may consist of people from different organisations.
- Project teams may be inter-disciplinary groups and are likely to lie outside the normal organisation hierarchies.
- The project team will be responsible for delivery of the project end product to some sponsor within or outside the organisation. The full benefit of any project will not become available until the project as been completed.

1.4.2 Project Classification:

In recent years more and more activities have been tackled on a project basis. Project teams and a project management approach have become common in most organisations. The basic approaches to project management remain the same regardless of the type of project being considered. You may find it useful to consider projects in relation to a number of major classifications:

a) Engineering and construction

The projects are concerned with producing a clear physical output, such as roads, bridges or buildings. The requirements of a project team are well defined in terms of skills and background, as are the main procedures that have to be undergone. Most of the problems which may confront the project team are likely to have occurred before and therefore their solution may be based upon past experiences.

b) Introduction of new systems

These projects would include computerisation projects and the introduction of new systems and procedures including financial systems. The nature and constitution of a project team may vary with the subject of the project, as different skills may be required and different end-users may be involved. Major projects involving a systems analysis approach may incorporate clearly defined procedures within an organisation.

c) Responding to deadlines and change

An example of responding to a deadline is the preparation of an annual report by a specified date. An increasing number of projects are concerned with designing organisational or environmental changes, involving developing new products and services.

1.4.3 Project Management Tools and techniques:

Project planning is at the heart of project management. One can't manage and control project activities if there is no plan. Without a plan, it is impossible to know if the correct activities are underway, if the available resources are adequate or of the project can be completed within the desired time. The plan becomes the roadmap that the project team members use to guide them through the project activities. Project management tools and techniques assist project managers and their teams in carrying out work in all nine knowledge areas. For example, some popular time- management tools and techniques include Gantt charts, project network diagrams, and critical path analysis. Table 1.1 lists some commonly used tools and techniques by knowledge area.

Knowledge Area	Tools & Techniques
Integration management	Project selection methods, project management methodologies, stakeholder analyses, project charters, project management plans, project management software, change requests, change control boards, project review meetings, lessons-learned reports
Scope management	Scope statements, work breakdown structures, mind maps, statements of work, requirements analyses, scope management plans, scope verification techniques, and scope change controls

Cost Management	Net present value, return on investment, payback analyses, earned value management, project portfolio management, cost estimates, cost management plans, cost baselines
Time management	Gantt charts, project network diagrams, critical-path analyses, crashing, fast tracking, schedule performance measurements
Human resource management	Motivation techniques, empathic listening, responsibility assignment matrices, project organizational charts, resource histograms, team building exercises
Quality management	Quality metrics, checklists, quality control charts, Pareto diagrams, fishbone diagrams, maturity models, statistical methods
Risk management	Risk management plans, risk registers, probability/impact matrices, risk rankings
Communication management	Communications management plans, kickoff meetings, conflict management, communications media selection, status and progress reports, virtual communications, templates, project Web sites
Procurement management	Make-or-buy analyses, contracts, requests for proposals or quotes, source selections, supplier evaluation matrices

Table 1.1 : Project Management Tools and Techniques

1.4.4 Project Success Factors:

The successful design, development, and implementation of information technology (IT) projects is a very difficult and complex process. However, although developing IT projects can be difficult, the reality is that a relatively small number of factors control the success or failure of every IT project, regardless of its size or complexity. The problem is not that the factors are unknown; it is that they seldom form an integral part of the IT development process.

Some of the factors that influence projects and may help them succeed are

- Executive Support
- User involvement
- Experienced project managers
- Limited scope
- Clear basic requirements

- Formal methodology
- Reliable estimates

1.5 INTRODUCTION TO AN OVERVIEW OF IT PROJECT MANAGEMENT

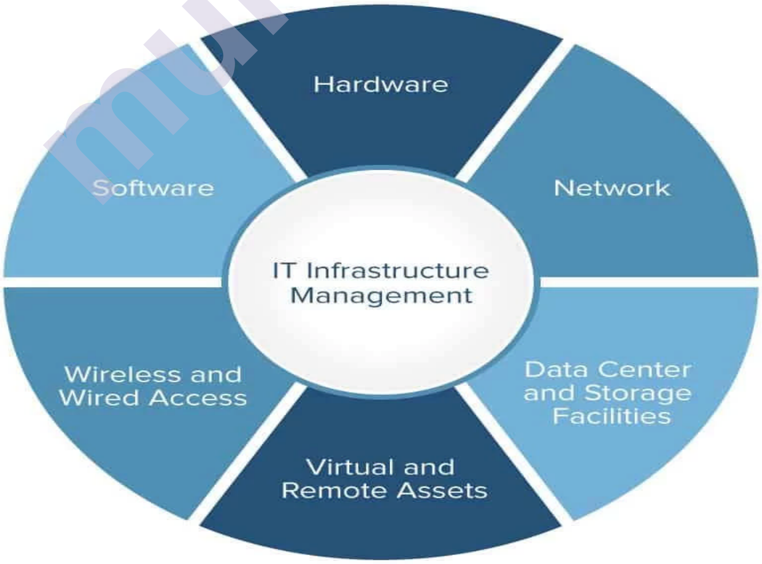
In IT, projects have become more complex as technologies rapidly change and end-users demand greater ease-of-use and flexibility. For an IT project manager to achieve their objectives, it is imperative that these initiatives are completed on time and on budget.

Discover what it means to manage IT projects, common challenges faced by IT project managers, and tips to make your next IT project a success. You'll also find helpful resources, like guides and free templates.

1.5.1 What Is IT Project Management?

IT project management (ITPM) is the process of managing the plan, organization, and accountability to achieve information technology goals. Since the reach of IT spans across most of a business or enterprise, the scope of these projects can be large and complex.

The magnitude of IT project management often means that it's more than just applying knowledge, aligning skills, and using regular tools and techniques to drive a project to completion. IT project managers deal with the challenges of interdependent integrations, rapid technology upgrades, and version changes that can occur throughout the project timeline.


Hardware
Network
Data Center and Storage Facilities
Virtual and Remote Assets
Wireless and Wired Access
Software
IT Infrastructure Management

1.5.2 Project Management Framework

- **Initiation:**

Project Managers work with other concerned stakeholders to evaluate and determine the values and feasibility of the project

- **Planning:**

  Create a project plan: team, timeline, activities, and resources budgeting. Designing required solutions to solve the customer's problem. Communicate project plan to all stakeholders.

- **Execution**:

  Team delivery of project plan. Test project implementation to ensure successful project integration. Monitor all aspect of project plan to ensure quality and delivery time. Establish training and modes of continues support for customers

- **Performance Control:**

  Project execution process gets evaluated in this phase by means of KPIs like project objectives, quality deliverables, cost monitoring, overall project performance, etc.

- **Closure**:

  The promised project deliverables are handed over to the client for validation. A final closure meeting is also held to discuss the overall experience and to close all project accounts.

1.5.3 Responsibilities of an IT Project Manager

Main Responsibilities of an IT Project Manager:

Today's IT project managers (IT PM) must be able to juggle a wide range of tasks and responsibilities. They must be able to handle firmware and software integrations, website construction, database storage and management, and also build complex and geographically diverse infrastructures and networks, all while planning for potential security and data risks.

Throughout their projects, IT PM is responsible for setting goals, communicating and motivating team members and stakeholders, identifying the right resources for each task, researching, managing change, performing needs assessment, and properly sequencing tasks.

Additional responsibilities of the IT project manager include the following:

- Project planning and overall management
- Promoting and achieving project support
- Ensuring overall capability with existing technology
- Minimizing duplicate work
- Utilizing team member skills
- Controlling costs and maintaining budgets

Challenges Faced by IT Project Managers:

The complexities and interdependencies of large-scale, long-term, diverse IT projects are among the most challenging issues of IT projects. Here are a few more top challenges faced by IT project managers:

- Making multiple assumptions when integrating different hardware, networks, and software to the existing system.
- Unclear expectations from the business, end-users, and stakeholders.
- Rapidly changing technology, leading to necessary mid-project upgrades that can affect timelines.
- Geographically diverse offices and remote work associated.

1.5.4 System view of a project

A system view of Project is to take a look into the scope of the project and to know how does that fit into the organization by analyzing the project using the following interrelated elements,

Programmed Objectives

- Rules, Regulations, Constraints and Policy Restrictions
- Management Control
- Inputs
- Implementation Process
- Output
- Outcome
- Impact
- Feedback

So by analyzing the above elements of the project, a project can be planned and managed.

1.5.5 Stakeholder Management

Who are Stakeholders?

Stakeholders are individuals who get impacted by the project. A Stakeholder can be a supporter and a resistor.

Project Stakeholder Management involves identification of stakeholders, analysis of their expectations and influences, development of appropriate strategies to work with the stakeholders and executing the process. Frequent communication is required with the stakeholders. Needs and expectations of the stakeholders to be understood. Managing conflicting interest and involving stakeholders in key project decisions and activities are also crucial. All of this forms a part of the stakeholder management process. The project manager is expected to possess the ability to identify the needs and influences of the stakeholders to manage them effectively.

Identify Stakeholders:

The process of identifying individuals who are impacted by the project is known as Identify Stakeholders Process. The project manager will be able to identify the appropriate focus of each stakeholder as an outcome of Identify Stakeholders process. Stakeholders can include the customers, sponsors, employees, management, government, and society as well. These stakeholders have a potential to exert positive or negative influence on the project deliverables.

Stakeholder needs are to be identified at an early stage of the project to ensure that all their requirements and voices are considered. The stakeholders can be classified on the basis of their interest in the project, the level of influence on the project outcome and their involvement. For the success of the project, the project manager needs to have a relationship that is cordial and extremely success oriented.

Stakeholders Process can receive inputs from:

Project Charter: Internal and external parties related to the project are identified using the project charter

Procurement Documents: The parties involved in a procurement contract are key project stakeholders

Enterprise Environmental Factors: Organizational culture, its structure, governmental regulations, trends, practices or habits of individuals represent enterprise environmental factors.

Organizational Process Assets: Stakeholder registers from previous projects, lessons learned are important inputs for identifying stakeholders

Stakeholder Analysis

A qualitative and quantitative analysis is required to systematically determine the interest of stakeholders throughout the project. The benefits of this analysis are:

Stakeholder interests can be identified

Stakeholder expectations can be identified

Another benefit includes identification of stakeholder relationships that can be leveraged to build partnerships with stakeholders to increase the probability of project success

Steps involved in stakeholder analysis process are:

Identification of potential stakeholders including their roles, departments, interests, knowledge, expectations, and influence levels.

Identify and analyze the potential impact each stakeholder could generate Classify the stakeholder's basis logical categories of potential impact Determine the likely reaction of these stakeholders to respond in various situations Plan the approach strategy to enhance their positive support and reduce negative influences

Multiple classification models are used for stakeholder analysis, but not limited to:

Power/Interest grid: Bifurcation of stakeholders based on their level of authority and their level of concern regarding project outcomes.

Power/Influence grid: Bifurcation of stakeholders based on their level of authority and their level of involvement in the project

Power/Impact grid: Bifurcation of stakeholders based on their level of authority and their level of impacting changes on project activities

Salience model: This model describes categories of stakeholders based on their power, urgency, and legitimacy.

Outputs of Identifying Stakeholders:

- Stakeholder register is updated with details such as:
- Stakeholder information
- Includes their name, organizational position, location, role in the project, business phone number, email address, etc
- Stakeholder requirements
- Key expectations, major requirements, involvement in the project etc
- Stakeholder Classification

1.5.6 Project Phases and the project life cycle

The Project Life Cycle (Phases):

The project manager and project team have one shared goal: to carry out the work of the project for the purpose of meeting the project's objectives. Every project has a beginning, a middle period during which activities move the project toward completion, and an ending (either successful or unsuccessful). A standard project typically has the following four major phases (each with its own agenda of tasks and issues): initiation, planning, implementation, and closure. Taken together, these phases represent the path a project takes from the beginning to its end and are generally referred to as the project "life cycle."

Initiation Phase:

During the first of these phases, the initiation phase, the project objective or need is identified; this can be a business problem or opportunity. An appropriate response to the need is documented in a business case with recommended solution options. A feasibility study is conducted to investigate whether each option addresses the project objective and a final recommended solution is determined. Issues of feasibility ("can we do the project?") and justification ("should we do the project?") are addressed.

Once the recommended solution is approved, a project is initiated to deliver the approved solution and a project manager is appointed. The major deliverables and the participating work groups are identified, and the project team begins to take shape. Approval is then sought by the project manager to move onto the detailed planning phase.

Planning Phase:

The next phase, the planning phase, is where the project solution is further developed in as much detail as possible and the steps necessary to meet the project's objective are planned. In this step, the team identifies all of the work to be done. The project's tasks and resource requirements are identified, along with the strategy for producing them. This is also referred to as "scope management." A project plan is created outlining the activities, tasks, dependencies, and timeframes. The project manager coordinates the preparation of a project budget by providing cost estimates for the labour, equipment, and materials costs. The budget is used to monitor and control cost expenditures during project implementation.

Once the project team has identified the work, prepared the schedule, and estimated the costs, the three fundamental components of the planning process are complete. This is an excellent time to identify and try to deal with anything that might pose a threat to the successful completion of the project. This is called risk management. In risk management, “high-threat” potential problems are identified along with the action that is to be taken on each high-threat potential problem, either to reduce the probability that the problem will occur or to reduce the impact on the project if it does occur. This is also a good time to identify all project stakeholders and establish a communication plan describing the information needed and the delivery method to be used to keep the stakeholders informed.

Finally, you will want to document a quality plan, providing quality targets, assurance, and control measures, along with an acceptance plan, listing the criteria to be met to gain customer acceptance. At this point, the project would have been planned in detail and is ready to be executed.

Implementation (Execution) Phase:

During the third phase, the implementation phase, the project plan is put into motion and the work of the project is performed. It is important to maintain control and communicate as needed during implementation. Progress is continuously monitored and appropriate adjustments are made and recorded as variances from the original plan. In any project, a project manager spends most of the time in this step. During project implementation, people are carrying out the tasks, and progress information is being reported through regular team meetings. The project manager uses this information to maintain control over the direction of the project by comparing the progress reports with the project plan to measure the performance of the project activities and take corrective action as needed. The first course of action should always be to bring the project back on course (i.e., to return it to the original plan). If that cannot happen, the team should record variations from the original plan and record and publish modifications to the plan. Throughout this step, project sponsors and other key stakeholders should be kept informed of the project’s status according to the agreed-on frequency and format of communication. The plan should be updated and published on a regular basis.

Status reports should always emphasize the anticipated end point in terms of cost, schedule, and quality of deliverables. Each project deliverable produced should be reviewed for quality and measured against the acceptance criteria. Once all of the deliverables have been produced and the customer has accepted the final solution, the project is ready for closure.

Closing Phase:

During the final closure, or completion phase, the emphasis is on releasing the final deliverables to the customer, handing over project documentation to the business, terminating supplier contracts, releasing project resources, and communicating the closure of the project to all stakeholders. The last remaining step is to conduct lessons-learned studies to examine what went well and what didn't. Through this type of analysis, the wisdom of experience is transferred back to the project organization, which will help future project teams.

Example: Project Phases on a Large Multinational Project:

A U.S. construction company won a contract to design and build the first copper mine in northern Argentina. There was no existing infrastructure for either the mining industry or large construction projects in this part of South America. During the initiation phase of the project, the project manager focused on defining and finding a project leadership team with the knowledge, skills, and experience to manage a large complex project in a remote area of the globe. The project team set up three offices. One was in Chile, where large mining construction project infrastructure existed. The other two were in Argentina. One was in Buenos Aries to establish relationships and Argentinean expertise, and the second was in Catamarca—the largest town close to the mine site. With offices in place, the project start-up team began developing procedures for getting work done, acquiring the appropriate permits, and developing relationships with Chilean and Argentine partners.

During the planning phase, the project team developed an integrated project schedule that coordinated the activities of the design, procurement, and construction teams. The project controls team also developed a detailed budget that enabled the project team to track project expenditures against the expected expenses. The project design team built on the conceptual design and developed detailed drawings for use by the procurement team. The procurement team used the drawings to begin ordering equipment and materials for the construction team; develop labour projections; refine the construction schedule; and set up the construction site. Although planning is a never-ending process on a project, the planning phase focused on developing sufficient details to allow various parts of the project team to coordinate their work and allow the project management team to make priority decisions.

The implementation phase represents the work done to meet the requirements of the scope of work and fulfil the charter. During the implementation phase, the project team accomplished the work defined in the plan and made adjustments when the project factors changed. Equipment and materials were delivered to the work site, labour was hired and trained, a construction site was built, and all the construction activities, from the arrival of the first dozer to the installation of the final light switch, were accomplished.

The closeout phase included turning over the newly constructed plant to the operations team of the client. A punch list of a few remaining construction items was developed and those items completed. The office in Catamarca was closed, the office in Buenos Aries archived all the project documents, and the Chilean office was already working on the next project. The accounting books were reconciled and closed, final reports written and distributed, and the project manager started on a new project.

1.6 SUMMARY

- IT project managers deal with the challenges of interdependent integrations, rapid technology upgrades, and version changes that can occur throughout the project timeline.
- IT project manager is responsible for setting goals, communicating and motivating team members and stakeholders, identifying the right resources for each task, researching, managing change, performing needs assessment, and properly sequencing tasks.
- Stakeholders have a potential to exert positive or negative influence on the project deliverables.
- Project Life Cycle contains initiation, planning, implementation, and closure.

❖❖❖❖

Module II

2

SOFTWARE PROCESS MODELS

Unit Structure

2.0 OBJECTIVES

After going through this unit, you will be able to learn:

- Different kind of approaches to built Software
- Software development approaches benefits and limitations.
- Need for Software Development Models

2.1 INTRODUCTION

Software Process Models were originally proposed to bring order to the chaos of software development. History has indicated that these conventional models have brought a certain amount of useful structure to software engineering work and have provided reasonably effective roadmap for software teams.

2.2 SOFTWARE PROCESS MODELS

A software process model (sometimes called a Software Development Life Cycle or SDLC model) is a simplified representation of a software process. Each process model represents a process from a particular perspective and thus only provides partial information about that process. For example, a process activity model shows the activities and their sequence but may not show the roles of the people involved in these activities. In this section, I introduce a number of very general process models (sometimes called process paradigms) and present these from an architectural perspective. That is, we see the framework of the process but not the details of process activities. These generic models are high-level, abstract descriptions of software processes that can be used to explain different approaches to software development. You can think of them as process frameworks that may be extended and adapted to create more specific software engineering processes.

2.2.1 Waterfall Model

Waterfall Model:

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

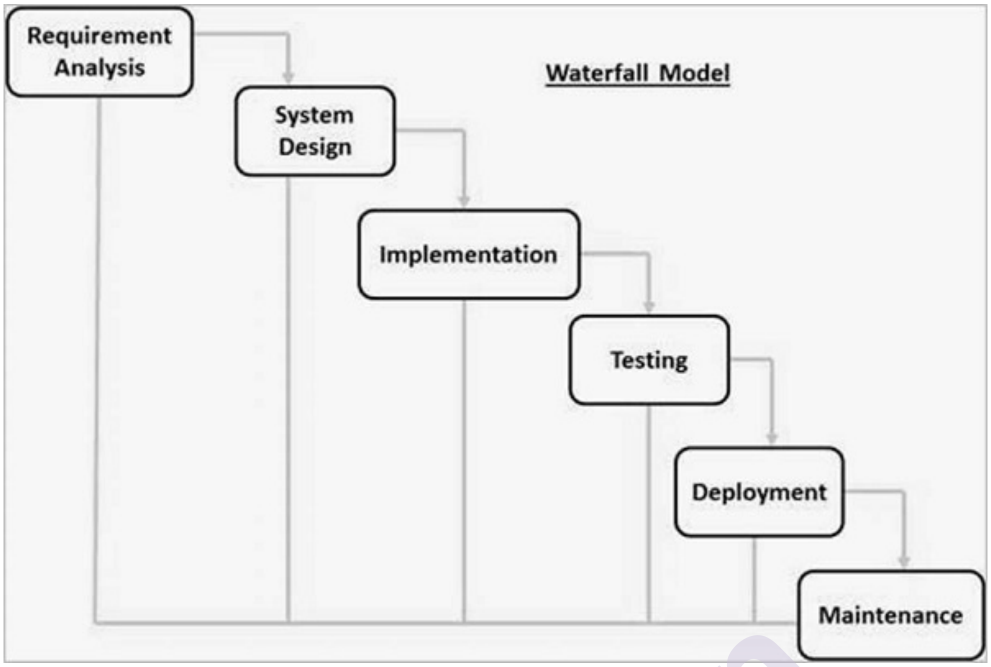
The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall Model – Design:

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are −

- Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

- System Design − the requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

- Implementation − with inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

- Integration and Testing − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- Deployment of system − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

- Maintenance − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

Waterfall Model – Application:

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model - Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model – Disadvantages:

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.

- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

2.2.2 Evolutionary process model:

Evolutionary model is a combination of Iterative and Incremental model of software development life cycle. Delivering your system in a big bang release, delivering it in incremental process over time is the action done in this model. Some initial requirements and architecture envisioning need to be done.

It is better for software products that have their feature sets redefined during development because of user feedback and other factors. The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users are able to get access to the product at the end of each cycle.

Feedback is provided by the users on the product for the planning stage of the next cycle and the development team responds, often by changing the product, plan or process. Therefore, the software product evolves with time.

All the models have the disadvantage that the duration of time from start of the project to the delivery time of a solution is very high. Evolutionary model solves this problem in a different approach.

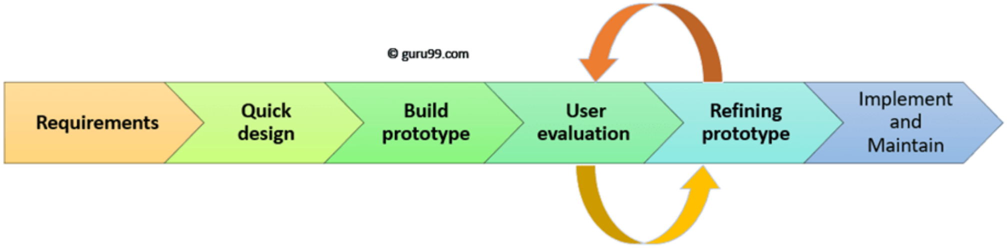
Evolutionary model suggests breaking down of work into smaller chunks, prioritizing them and then delivering those chunks to the customer one by one. The number of chunks is huge and is the number of deliveries made to the customer. The main advantage is that the customer’s confidence increases as he constantly gets quantifiable goods or services from the beginning of the project to verify and validate his requirements. The model allows for changing requirements as well as all work in broken down into maintainable work chunks.

2.2.2.1 Prototype Model

Prototype methodology is defined as a Software Development model in which a prototype is built, tests, and then reworked when needed until an acceptable prototype is achieved. It also creates a base to produce the final system.

Software prototyping model works best in scenarios where the project's requirement are not known. It is an iterative, trial, and error method which take place between the developer and the client.

Prototyping Model Phases



Prototyping Model has following six SDLC phases as follow:

Step 1: Requirements gathering and analysis

A prototyping model starts with requirement analysis. In this phase, the requirements of the system are defined in detail. During the process, the users of the system are interviewed to know what their expectation from the system is.

Step 2: Quick design

The second phase is a preliminary design or a quick design. In this stage, a simple design of the system is created. However, it is not a complete design. It gives a brief idea of the system to the user. The quick design helps in developing the prototype.

Step 3: Build a Prototype

In this phase, an actual prototype is designed based on the information gathered from quick design. It is a small working model of the required system.

Step 4: Initial user evaluation

In this stage, the proposed system is presented to the client for an initial evaluation. It helps to find out the strength and weakness of the working model. Comment and suggestion are collected from the customer and provided to the developer.

Step 5: Refining prototype

If the user is not happy with the current prototype, you need to refine the prototype according to the user's feedback and suggestions.

This phase will not over until all the requirements specified by the user are met. Once the user is satisfied with the developed prototype, a final system is developed based on the approved final prototype.

Step 6: Implement Product and Maintain

Once the final system is developed based on the final prototype, it is thoroughly tested and deployed to production. The system undergoes routine maintenance for minimizing downtime and prevents large-scale failures.

Four types of Prototyping models are:

- Rapid Throwaway prototypes
- Evolutionary prototype
- Incremental prototype
- Extreme prototype

Rapid Throwaway Prototype:

Rapid throwaway is based on the preliminary requirement. It is quickly developed to show how the requirement will look visually. The customer's feedback helps drives changes to the requirement, and the prototype is again created until the requirement is base lined. In this method, a developed prototype will be discarded and will not be a part of the ultimately accepted prototype. This technique is useful for exploring ideas and getting instant feedback for customer requirements.

Evolutionary Prototyping:

Here, the prototype developed is incrementally refined based on customer's feedback until it is finally accepted. It helps you to save time as well as effort. That's because developing a prototype from scratch for every interaction of the process can sometimes be very frustrating.

This model is helpful for a project which uses a new technology that is not well understood. It is also used for a complex project where every functionality must be checked once. It is helpful when the requirement is not stable or not understood clearly at the initial stage.

Incremental Prototyping:

In incremental Prototyping, the final product is decimated into different small prototypes and developed individually. Eventually, the different prototypes are merged into a single product. This method is helpful to reduce the feedback time between the user and the application development team.

Extreme Prototyping:

Extreme prototyping method is mostly used for web development. It is consists of three sequential phases.

Basic prototype with the entire existing page is present in the HTML format. You can simulate data process using a prototype services layer. The services are implemented and integrated into the final prototype. Best practices of Prototyping. Here, are a few things which you should watch for during the prototyping process:

You should use Prototyping when the requirements are unclear. It is important to perform planned and controlled Prototyping. Regular meetings are vital to keep the project on time and avoid costly delays. The users and the designers should be aware of the prototyping issues and pitfalls. At a very early stage, you need to approve a prototype and only then allow the team to move to the next step.

In software prototyping method, you should never be afraid to change earlier decisions if new ideas need to be deployed. You should select the appropriate step size for each version. Implement important features early on so that if you run out of the time, you still have a worthwhile system

Advantages of the Prototyping Model

Here, are important pros/benefits of using Prototyping models:

- Users are actively involved in development. Therefore, errors can be detected in the initial stage of the software development process.
- Missing functionality can be identified, which helps to reduce the risk of failure as Prototyping is also considered as a risk reduction activity.
- Helps team member to communicate effectively.
- Customer satisfaction exists because the customer can feel the product at a very early stage. There will be hardly any chance of software rejection.
- Quicker user feedback helps you to achieve better software development solutions. It allows the client to compare if the software code matches the software specification.
- It helps you to find out the missing functionality in the system. It also identifies the complex or difficult functions that encourage innovation and flexible designing.
- It is a straightforward model, so it is easy to understand. No need for specialized experts to build the model.
- The prototype serves as a basis for deriving a system specification.
- The prototype helps to gain a better understanding of the customer's needs. Prototypes can be changed and even discarded.
- Prototype also serves as the basis for operational specifications.
- Prototypes may offer early training for future users of the software system.

Disadvantages of the Prototyping Model:

- Prototyping is a slow and time taking process.
- The cost of developing a prototype is a total waste as the prototype is ultimately thrown away.
- Prototyping may encourage excessive change requests.
- Sometimes customers may not be willing to participate in the iteration cycle for the longer time duration.
- There may be far too many variations in software requirements when each time the prototype is evaluated by the customer.
- The poor documentation because the requirements of the customers are changing.
- It is very difficult for software developers to accommodate all the changes demanded by the clients.

- After seeing an early prototype model, the customers may think that the actual product will be delivered to him soon.
- The client may lose interest in the final product when he or she is not happy with the initial prototype.
- Developers who want to build prototypes quickly may end up building sub-standard development solutions.

2.2.2.2 Spiral Model

Spiral Model:

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

Spiral Model – Design:

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

Identification:

This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase. This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

Design:

The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

Construct or Build:

The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback. Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

Evaluation and Risk Analysis:

Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback. The following illustration is a representation of the Spiral Model, listing the activities in each phase. Based on the customer evaluation, the software development process enters the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

Spiral Model Application:

The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e. learning with maturity which involves minimum risk for the customer as well as the development firms.

The following pointers explain the typical uses of a Spiral Model −

- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which are usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

Spiral Model - Pros and Cons:

The advantage of spiral lifecycle model is that it allows elements of the product to be added in, when they become available or known. This assures that there is no conflict with previous requirements and design.

This method is consistent with approaches that have multiple software builds and releases which allows making an orderly transition to a maintenance activity. Another positive aspect of this method is that the spiral model forces an early user involvement in the system development effort.

On the other side, it takes a very strict management to complete such products and there is a risk of running the spiral in an indefinite loop. So, the discipline of change and the extent of taking change requests is very important to develop and deploy the product successfully.

The advantages of the Spiral SDLC Model are as follows −

- Changing requirements can be accommodated.
- Allows extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

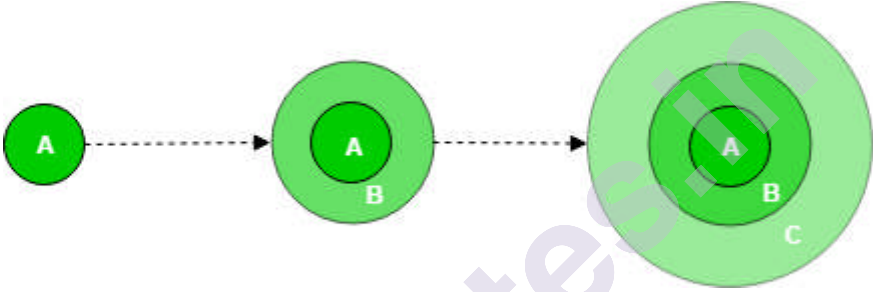
The disadvantages of the Spiral SDLC Model are as follows −

- Management is more complex.
- End of the project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go on indefinitely.
- Large number of intermediate stages requires excessive documentation.

2.2.3 Incremental Process Model

Incremental process model is also known as successive version model.

First, a simple working system implementing only a few basic features is built and then that is delivered to the customer. Then thereafter many successive iterations/ versions are implemented and delivered to the customer until the desired system is released.



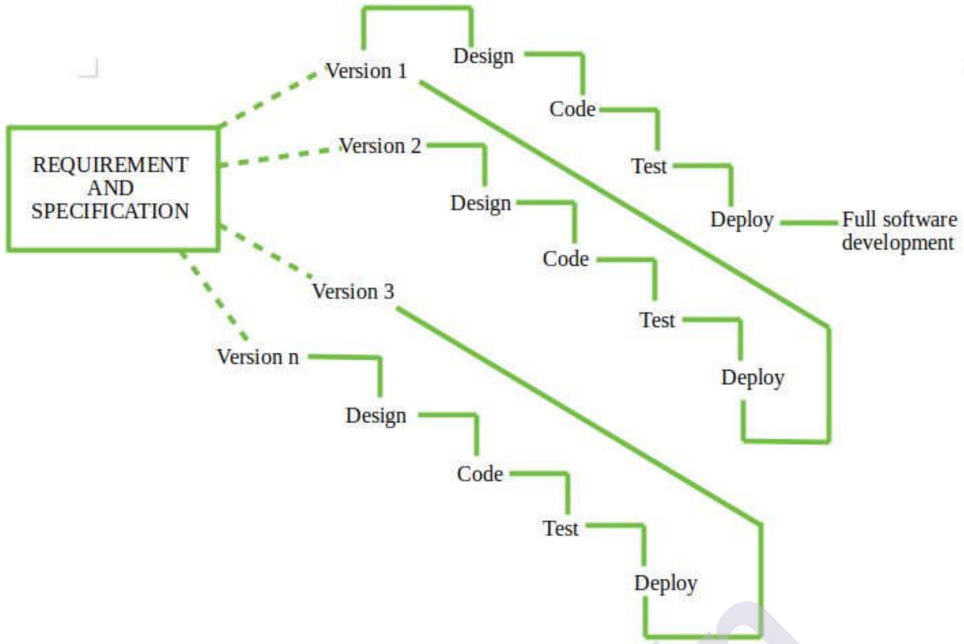
A, B, C are modules of Software Product that are incrementally developed and delivered.

Life cycle activities –

Requirements of Software are first broken down into several modules that can be incrementally constructed and delivered. At any time, the plan is made just for the next increment and not for any kind of long term plans. Therefore, it is easier to modify the version as per the need of the customer. Development Team first undertakes to develop core features (these do not need services from other features) of the system.

Once the core features are fully developed, then these are refined to increase levels of capabilities by adding new functions in Successive versions. Each incremental version is usually developed using an iterative waterfall model of development.

As each successive version of the software is constructed and delivered, now the feedback of the Customer is to be taken and these were then incorporated in the next version. Each version of the software has more additional features over the previous ones.



After Requirements gathering and specification, requirements are then spitted into several different versions starting with version-1, in each successive increment, next version is constructed and then deployed at the customer site. After the last version (version n), it is now deployed at the client site.

Types of Incremental model –

1. Staged Delivery Model – Construction of only one part of the project at a time.
2. Parallel Development Model – Different subsystems are developed at the same time. It can decrease the calendar time needed for the development, i.e. TTM (Time to Market), if enough Resources are available.

When to use this –

1. Funding Schedule, Risk, Program Complexity, or need for early realization of benefits.
2. When Requirements are known up-front.
3. When Projects having lengthy developments schedules.
4. Projects with new Technology.

Advantages –

- Error Reduction (core modules are used by the customer from the beginning of the phase and then these are tested thoroughly)
- Uses divide and conquer for breakdown of tasks.
- Lowers initial delivery cost.
- Incremental Resource Deployment.

Disadvantages –

- Requires good planning and design.
- Total cost is not lower.
- Well defined module interfaces are required.

2.2.3.1 Iterative Approach

In an Iterative Incremental model, initially, a partial implementation of a total system is constructed so that it will be in a deliverable state. Increased functionality is added. Defects, if any, from the prior delivery are fixed and the working product is delivered. The process is repeated until the entire product development is completed. The repetitions of these processes are called iterations. At the end of every iteration, a product increment is delivered.

Iterative Incremental Model – Strengths

The advantages or strengths of Iterative Incremental model are −

- You can develop prioritized requirements first.
- Initial product delivery is faster.
- Customers gets important functionality early.
- Lowers initial delivery cost.
- Each release is a product increment, so that the customer will have a working product at hand all the time.
- Customer can provide feedback to each product increment, thus avoiding surprises at the end of development.
- Requirements changes can be easily accommodated.

Iterative Incremental Model – Weaknesses

The disadvantages of the Iterative Incremental model are −

- Requires effective planning of iterations.
- Requires efficient design to ensure inclusion of the required functionality and provision for changes later.
- Requires early definition of a complete and fully functional system to allow the definition of increments.
- Well-defined module interfaces are required, as some are developed long before others are developed.
- Total cost of the complete system is not lower.

When to Use Iterative Incremental Model?

- Iterative Incremental model can be used when −
- Most of the requirements are known up-front but are expected to evolve over time.
- The requirements are prioritized.
- There is a need to get the basic functionality delivered fast.
- A project has lengthy development schedules.
- A project has new technology.
- The domain is new to the team.

2.2.3.2 Rapid Application Development

RAD (Rapid Application Development) model:

The RAD (Rapid Application Development) model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product.

Rapid Application Development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

What is RAD?

Rapid application development is a software development methodology that uses minimal planning in favour of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product. In the RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process. RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype. The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable.

RAD Model Design

RAD model distributes the analysis, design, build and test phases into a series of short, iterative development cycles.

Following are the various phases of the RAD Model −
Business Modelling:

The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels. A complete business analysis is performed to find the vital information for business, how it can be obtained, how and when is the information processed and what are the factors driving successful flow of information.

Data Modelling:

The information gathered in the Business Modeling phase is reviewed and analyzed to form sets of data objects vital for the business. The attributes of all data sets is identified and defined. The relation between these data objects are established and defined in detail in relevance to the business model.

Process Modelling:

The data object sets defined in the Data Modeling phase are converted to establish the business information flow needed to achieve specific business objectives as per the business model. The process model for any changes or enhancements to the data object sets is defined in this phase. Process descriptions for adding, deleting, retrieving or modifying a data object are given.

Application Generation:

The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.

Testing and Turnover:

The overall testing time is reduced in the RAD model as the prototypes are independently tested during every iteration. However, the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues.

The following illustration describes the RAD Model in detail.

RAD Model Vs Traditional SDLC:

The traditional SDLC follows a rigid process models with high emphasis on requirement analysis and gathering before the coding starts. It puts pressure on the customer to sign off the requirements before the project starts and the customer doesn't get the feel of the product as there is no working build available for a long time. The customer may need some changes after he gets to see the software. However, the change process is quite rigid and it may not be feasible to incorporate major changes in the product in the traditional SDLC. The RAD model focuses on iterative and incremental delivery of working models to the customer. This results in rapid delivery to the customer and customer involvement during the complete development cycle of product reducing the risk of non-conformance with the actual user requirements.

RAD Model – Application:

RAD model can be applied successfully to the projects in which clear modularization is possible. If the project cannot be broken into modules, RAD may fail.

The following pointers describe the typical scenarios where RAD can be used −

- RAD should be used only when a system can be modularized to be delivered in an incremental manner.
- It should be used if there is a high availability of designers for modeling.
- It should be used only if the budget permits use of automated code generating tools.
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
- Should be used where the requirements change during the project and working prototypes are to be presented to customer in small iterations of 2-3 months.

RAD Model - Pros and Cons

RAD model enables rapid delivery as it reduces the overall development time due to the reusability of the components and parallel development. RAD works well only if high skilled engineers are available and the customer is also committed to achieve the targeted prototype in the given time frame. If there is commitment lacking on either side the model may fail.

The advantages of the RAD Model are as follows −

- Changing requirements can be accommodated.
- Progress can be measured.
- Iteration time can be short with use of powerful RAD tools.
- Productivity with fewer people in a short time.
- Reduced development time.
- Increases reusability of components.
- Quick initial reviews occur.
- Encourages customer feedback.
- Integration from very beginning solves a lot of integration issues.

The disadvantages of the RAD Model are as follows −

- Dependency on technically strong team members for identifying business requirements.
- Only system that can be modularized can be built using RAD.
- Requires highly skilled developers/designers.
- High dependency on modelling skills.
- Inapplicable to cheaper projects as cost of modelling and automated code generation is very high.
- Management complexity is more.
- Suitable for systems that are component based and scalable.
- Requires user involvement throughout the life cycle.
- Suitable for project requiring shorter development times.

2.2.3.3 Joint Application Development Model

Collaboration and then building software is the key power which drives technology and its innovation. JAD is a model for software development that augments the stakeholders' association in cycles of software development. Its life cycle has been adopted for areas of dynamic software development method. It collects business and system requirements while building a new information system for any organization or enterprise. In this chapter, you will learn about the JAD model in detail.

JAD (Joint Application Development) Approach:

JAD (Joint Application Development) is a software development approach which engages the client and/or the end users for designing and developing the system. This model was designed and put forward by Dr. Chuck Morris and Dr. Tony Crawford of IBM, who propose this model in the late 1970s. As compared to other primitive SDLC model, Joint Application Development model leads to faster progression of the system development which has better client approval.

This model furthermore, is vast when it comes to agile delivery wherein the software products need to be developed as well as shipped in short iterations depending on agreements among the industrial as well as industry stakeholders which are termed as Minimum Viable Product (MVP).

Phases of JAD Model: Since you have become familiar with the JAD concept, it is time to know about its phases and how the model's design and development approach works:

Define Specific Objectives: The facilitator, in partnership with stakeholders, set all the objectives as well as a list of items which is then distributed to other developers and participants to understand and review. This objective contains elements like the scope of this projected system, its potential outcome, technical specification required, etc.

Session Preparation: The facilitator is solely responsible for this preparation where all relevant data is collected and sent to other members before time. For better insight, research carried out to know about the system requirement better and gather all the necessary information for development.

Session Conduct: Here the facilitator is accountable to identify those issues which have to be working out for making the system error-free. Here the facilitator will serve as a participant but will not have a say regarding any information.

Documentation: After the product is developed, the records and published documents are put forward into the meeting so that the stakeholders and consumers can approve it through the meeting.

Benefits of using JAD Model:

- **Improved Delivery Time:** The time required for developing a product using JAD model is lesser and efficient than that of other traditional models.

- **Cost Reduction:** Efficiently analyzing the requirements and facts with business executives and stakeholders will make less effort to develop the system and hence less cost will be required for the entire development process.

- **Better Understanding:** Since the entire requirement is analyzed by business executives, followed by a cautious choice of developers and team member who can professionally interact with each other better usually helps in understanding the product development better.

- **Improved Quality:** Since all the key decision makers and stakeholders of the project are involved in the development of the project so there is the least chance of error and hence the product quality becomes better and more accurate.

2.2.3.4 Concurrent Development Model

` The concurrent development model

- The concurrent development model is called as concurrent model.
- The communication activity has completed in the first iteration and exits in the awaiting changes state.
- The modelling activity completed its initial communication and then goes to the underdevelopment state.
- If the customer specifies the change in the requirement, then the modelling activity moves from the under development state into the awaiting change state.
- The concurrent process model activities moving from one state to another state

.

Advantages of the concurrent development model

- This model is applicable to all types of software development processes.
- It is easy to understand and use.
- It gives immediate feedback from testing.
- It provides an accurate picture of the current state of a project.

Disadvantages of the concurrent development model

- It needs better communication between the team members. This may not be achieved all the time.
- It requires remembering the status of the different activities.

2.2.4 Agile Development

Agile is a time-bound, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver all at once.

Why Agile?

Technology in this current era is progressing faster than ever, enforcing the global software companies to work in a fast-paced changing environment. Because these businesses are operating in an ever-changing environment, it is impossible to gather a complete and exhaustive set of software requirements. Without these requirements, it becomes practically hard for any conventional software model to work.

The conventional software models such as Waterfall Model that depends on completely specifying the requirements, designing, and testing the system are not geared towards rapid software development. As a consequence, a conventional software development model fails to deliver the required product.

This is where agile software development comes to the rescue. It was specially designed to curate the needs of the rapidly changing environment by embracing the idea of incremental development and developing the actual final product.

Let’s now read about the on which the Agile has laid its foundation:

Principles:

1. Highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. It welcomes changing requirements, even late in development.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shortest timescale.
4. Build projects around motivated individuals. Give them the environment and the support they need, and trust them to get the job done.
5. Working software is the primary measure of progress.
6. Simplicity the art of maximizing the amount of work not done is essential.
7. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Development in Agile: Let’s see a brief overview of how development occurs in agile philosophy.

- In Agile development, Design and Implementation are considered to be the central activities in the software process.
- Design and Implementation phase also incorporate other activities such as requirements elicitation and testing into it.
- In an agile approach, iteration occurs across activities. Therefore, the requirements and the design are developed together, rather than separately.
- The allocation of requirements and the design planning and development as executed in a series of increments. In contrast with the conventional model, where requirements gathering needs to be completed in order to proceed to the design and development phase, it gives Agile development an extra level of flexibility.
- An agile process focuses more on code development rather than documentation.

Example: Let’s go through an example to understand clearly about how agile actually works.

A Software company named ABC wants to make a new web browser for the latest release of its operating system. The deadline for the task is 10 months. The company’s head assigned two teams named Team A and Team B for this task. In order to motivate the teams, the company head says that the first team to develop the browser would be given a salary hike and a one week full sponsored travel plan. With the dreams of their wild travel fantasies, the two teams set out on the journey of the web browser. The team A decided to play by the book and decided to choose the Waterfall model for the development. Team B after a heavy discussion decided to take a leap of faith and choose Agile as their development model.

The Development plan of the Team A is as follows:

- Requirement analysis and Gathering – 1.5 Months
- Design of System – 2 Months
- Coding phase – 4 Months
- System Integration and Testing – 2 Months
- User Acceptance Testing – 5 Weeks

The Development plan for the Team B is as follows:

- Since this was an Agile, the project was broken up into several iterations.
- The iterations are all of the same time duration.
- At the end of each iteration, a working product with a new feature has to be delivered.
- Instead of Spending 1.5 months on requirements gathering, They will decide the core features that are required in the product and decide which of these features can be developed in the first iteration.
- Any remaining features that cannot be delivered in the first iteration will be delivered in the next subsequent iteration, based in the priority

At the end of the first iterations, the team will deliver working software with the core basic features.

Both the team have put their best efforts to get the product to a complete stage. But then out of blue due to the rapidly changing environment, the company's head come up with an entirely new set of features and want to be implemented as quickly as possible and wanted to push out a working model in 2 days. Team A was now in a fix, they were still in their design phase and did not yet started coding and they had no working model to display. And moreover, it was practically impossible for them to implement new features since waterfall model there is not reverting back to the old phase once you proceed to the next stage that means they would have to start from the square one again. That would incur them heavy cost and a lot of overtime. Team B was ahead of Team A in a lot of aspects, all thanks to Agile Development. They also had the working product with most of the core requirement since the first increment. And it was a piece of cake for them to add the new requirements. All they had to do is schedule these requirements for the next increment and then implement them.

Advantages:

- Deployment of software is quicker and thus helps in increasing the trust of the customer.
- Can better adapt to rapidly changing requirements and respond faster.
- Helps in getting immediate feedback which can be used to improve the software in the next increment.
- People – Not Process. People and interactions are given a higher priority rather than process and tools.
- Continuous attention to technical excellence and good design.

Disadvantages:

- In case of large software projects, it is difficult to assess the effort required at the initial stages of the software development life cycle.
- The Agile Development is more code focused and produces less documentation.
- Agile development is heavily depended on the inputs of the customer. If the customer has ambiguity in his vision of the final outcome, it is highly likely for the project to get off track.
- Face to Face communication is harder in large-scale organizations.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it's a difficult situation for new programmers to adapt to the environment.

Agile is a framework which defines how the software development needs to be carried on. Agile is not a single method, it represents the various collection of methods and practices that follow the value statements provided in the manifesto. Agile methods and practices do not promise to solve every problem present in the software industry (No Software model ever can). But they sure help to establish a culture and environment where solutions emerge.

2.2.4.1 Extreme Programming

Extreme programming (XP) is one of the most important software development framework of Agile models. It is used to improve software quality and responsive to customer requirements. The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

Good practices needs to practiced extreme programming: Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

- **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their works between them every hour.

- **Testing**: Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach test cases are written even before any code is written.

- **Incremental development**: Incremental development is very good because customer feedback is gained and based on this development team come up with new increments every few days after each iteration.

- **Simplicity**: Simplicity makes it easier to develop good quality code as well as to test and debug it.

- **Design**: Good quality design is important to develop a good quality software. So, everybody should design daily.

- **Integration testing**: It helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

Basic principles of Extreme programming: XP is based on the frequent iteration through which the developers implement User Stories. User stories are simple and informal statements of the customer about the functionalities needed. A User story is a conventional description by the user about a feature of the required system. It does not mention finer details such as the different scenarios that can occur. On the basis of User stories, the project team proposes Metaphors. Metaphors are a common vision of how the system would work. The development team may decide to build a Spike for some feature. A Spike is a very simple program that is constructed to explore the suitability of a solution being proposed. It can be considered similar to a prototype. Some of the basic activities that are followed during software development by using XP model are given below:

- **Coding**: The concept of coding which is used in XP model is slightly different from traditional coding. Here, coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system and choosing among several alternative solutions.

- **Testing**: XP model gives high importance on testing and considers it be the primary factor to develop a fault-free software.

- **Listening**: The developers needs to carefully listen to the customers if they have to develop a good quality software. Sometimes programmers may not have the depth knowledge of the system to be developed. So, it is desirable for the programmers to understand properly the functionality of the system and they have to listen to the customers.

- **Designing**: Without a proper design, a system implementation becomes too complex and very difficult to understand the solution, thus it makes maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.

- **Feedback**: One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.

- **Simplicity**: The main principle of the XP model is to develop a simple system that will work efficiently in present time, rather than trying to build something that would take time and it may never be used. It focuses on some specific features that are immediately needed, rather than engaging time and effort on speculations of future requirements.

Applications of Extreme Programming (XP): Some of the projects that are suitable to develop using XP model are given below:

- **Small projects**: XP model is very useful in small projects consisting of small teams as face to face meeting is easier to achieve.

- **Projects involving new technology or Research projects:** This type of projects faces changing of requirements rapidly and technical problems. So XP model is used to complete this type of projects.

1.2.4.2 Scrum

What is Scrum Project Management?

Scrum is an agile project management methodology or framework used primarily for software development projects with the goal of delivering new software capability every 2-4 weeks. It is one of the approaches that influenced the Agile Manifesto, which articulates a set of values and principles to guide decisions on how to develop higher-quality software faster.

Who Uses Agile Scrum Methodology?

Scrum is widely used by software development teams. In fact it's the most popular agile methodology. According to the 12th annual State of Agile report, 70% of software teams use Scrum or a Scrum hybrid. However, Scrum has spread to other business functions including IT and marketing where there are projects that must move forward in the presence of complexity and ambiguity. Leadership teams are also basing their agile management practices on Scrum, often combining it with lean and Kanban practices (subgroups of agile project management).

What is Scrum in Relation to Agile Project Management?

Scrum is a sub-group of agile:

- Agile is a set of values and principles that describe a group's day-to-day interactions and activities. Agile itself is not prescriptive or specific.

- The Scrum methodology follows the values and principles of agile, but includes further definitions and specifications, especially regarding certain software development practices.

Although developed for agile software development, agile Scrum became the preferred framework for agile project management in general and is sometimes simply referred to as Scrum project management or Scrum development.

What is the Benefits Received from the Scrum Methodology?

Organizations that have adopted agile Scrum have experienced:

- Higher productivity
- Better-quality products
- Reduced time to market
- Improved stakeholder satisfaction
- Better team dynamics
- Happier employees

2.3 SUMMARY

- Software development using Waterfall Model approach.
- Evolutionary Process Model allows for changing requirements as well as all work in broken down into maintainable work chunks.
- In Prototype Model, prototype is built, tests, and then reworked when needed until an acceptable prototype is achieved.
- Spiral Model allows incremental releases of the product or incremental refinement through each iteration around the spiral.
- Incremental process model is a successive version model.
- Agile is a time-bound, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver all at once.

❖❖❖❖

Module III

3

SOFTWARE REQUIREMENT ANALYSIS AND SPECIFICATION

Unit Structure

3.0 OBJECTIVES

The objective of this chapter is to introduce software requirements and to explain the processes involved in discovering and documenting these requirements. When you have read the chapter, you will:

- Understand the concepts of user and system requirements and why these requirements should be written in different ways;
- Understand the differences between functional and non-functional software requirements;
- Understand the main requirements engineering activities of elicitation, analysis, and validation, and the relationships between these activities;
- Understand why requirements management is necessary and how it supports other requirements engineering activities.

3.1 INTRODUCTION

The requirements for a system are the descriptions of the services that a system should provide and the constraints on its operation. These requirements reflect the needs of customers for a system that serves a certain purpose such as controlling a device, placing an order, or finding information. The process of finding out, analyzing, documenting and checking these services and constraints is called requirements engineering (RE).

The term requirement is not used consistently in the software industry. In some cases, a requirement is simply a high-level, abstract statement of a service that a system should provide or a constraint on a system. At the other extreme, it is a detailed, formal definition of a system function.

3.2 TYPES OF REQUIREMENT

Software system requirements are often classified as functional or non-functional requirements:

1. **Functional requirements:**
   These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do.

2. **Non-functional requirements:**
   These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, and constraints imposed by standards. Non-functional requirements often apply to the system as a whole rather than individual system features or services.

In reality, the distinction between different types of requirements is not as clear-cut as these simple definitions suggest. A user requirement concerned with security, such as a statement limiting access to authorized users, may appear to be a non-functional requirement. However, when developed in more detail, this requirement may generate other requirements that are clearly functional, such as the need to include user authentication facilities in the system. This shows that requirements are not independent and that one requirement often generates or constrains other requirements. The system requirements therefore do not just specify the services or the features of the system that are required; they also specify the necessary functionality to ensure that these services/features are delivered effectively.

3.2.1 Functional Requirement

The functional requirements for a system describe what the system should do. These requirements depend on the type of software being developed, the expected users of the software, and the general approach taken by the organization when writing requirements. When expressed as user requirements, functional requirements should be written in natural language so that system users and managers can understand them. Functional system requirements expand the user requirements and are written for system developers. They should describe the system functions, their inputs and outputs, and exceptions in detail.

Functional system requirements vary from general requirements covering what the system should do to very specific requirements reflecting local ways of working or an organization's existing systems. For example, here are examples of functional requirements for the Mentcare system, used to maintain information about patients receiving treatment for mental health problems:

1. A user shall be able to search the appointments lists for all clinics.
2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
3. Each staff member using the system shall be uniquely identified by his or her eight-digit employee number.

These user requirements define specific functionality that should be included in the system. The requirements show that functional requirements may be written at different levels of detail (contrast requirements 1 and 3).

Functional requirements, as the name suggests, have traditionally focused on what the system should do. However, if an organization decides that an existing off-the-shelf system software product can meet its needs, then there is very little point in developing a detailed functional specification. In such cases, the focus should be on the development of information requirements that specify the information needed for people to do their work. Information requirements specify the information needed and how it is to be delivered and organized. Therefore, an information requirement for the Mentcare system might specify what information is to be included in the list of patients expected for appointments that day.

Imprecision in the requirements specification can lead to disputes between customers and software developers. It is natural for a system developer to interpret an ambiguous requirement in a way that simplifies its implementation. Often, however, this is not what the customer wants. New requirements have to be established and changes made to the system. Of course, this delays system delivery and increases costs.

For example, the first Mentcare system requirement in the above list states that a user shall be able to search the appointments lists for all clinics. The rationale for this requirement is that patients with mental health problems are sometimes confused. They may have an appointment at one clinic but actually go to a different clinic. If they have an appointment, they will be recorded as having attended, regardless of the clinic.

A medical staff member specifying a search requirement may expect "search" to mean that, given a patient name, the system looks for that name in all appointments at all clinics. However, this is not explicit in the requirement. System developers may interpret the requirement so that it is easier to implement. Their search function may require the user to choose a clinic and then carry out the search of the patients who attended that clinic. This involves more user input and so takes longer to complete the search.

Ideally, the functional requirements specification of a system should be both complete and consistent. Completeness means that all services and information required by the user should be defined. Consistency means that requirements should not be contradictory.

In practice, it is only possible to achieve requirements consistency and completeness for very small software systems. One reason is that it is easy to make mistakes and omissions when writing specifications for large, complex systems. Another reason is that large systems have many stakeholders, with different backgrounds and expectations. Stakeholders are likely to have different—and often inconsistent— needs. These inconsistencies may not be obvious when the requirements are originally specified, and the inconsistent requirements may only be discovered after deeper analysis or during system development.

3.2.2 Non-Functional Requirement

Non-functional requirements, as the name suggests, are requirements that are not directly concerned with the specific services delivered by the system to its users. These non-functional requirements usually specify or constrain characteristics of the system as a whole. They may relate to emergent system properties such as reliability, response time, and memory use. Alternatively, they may define constraints on the system implementation, such as the capabilities of I/O devices or the data representations used in interfaces with other systems.

Non-functional requirements are often more critical than individual functional requirements. System users can usually find ways to work around a system function that doesn't really meet their needs. However, failing to meet a non-functional requirement can mean that the whole system is unusable. For example, if an aircraft system does not meet its reliability requirements, it will not be certified as safe for operation; if an embedded control system fails to meet its performance requirements,

The control functions will not operate correctly.

While it is often possible to identify which system components implement specific functional requirements (e.g., there may be formatting components that implement reporting requirements), this is often more difficult with non-functional requirements. The implementation of these requirements may be spread throughout the system, for two reasons:

1. Non-functional requirements may affect the overall architecture of a system rather than the individual components. For example, to ensure that performance requirements are met in an embedded system, you may have to organize the system to minimize communications between components.

2. An individual non-functional requirement, such as a security requirement, may generate several, related functional requirements that define new system services that are required if the non-functional requirement is to be implemented. In addition, it may also generate requirements that constrain existing requirements; for example, it may limit access to information in the system.

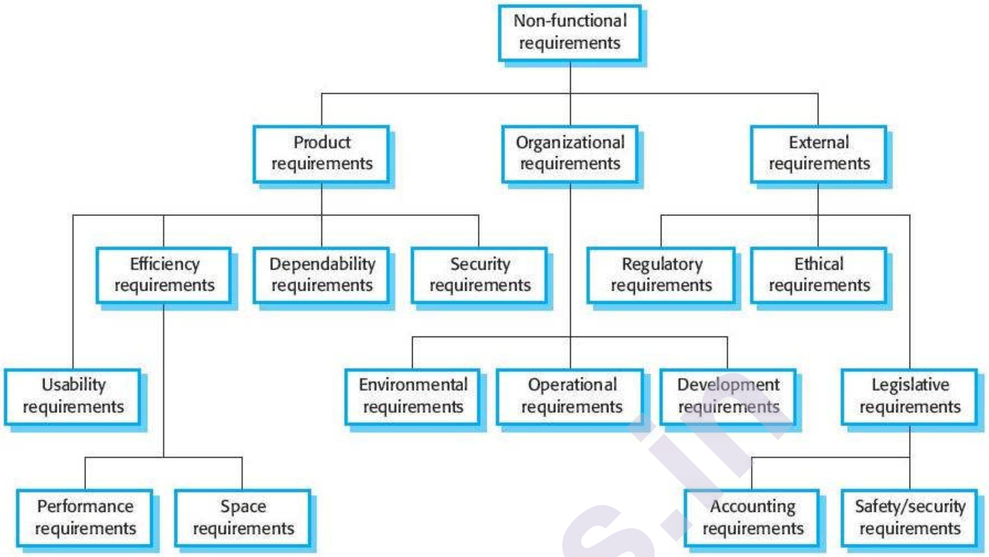


Figure: Types of Non-Functional Requirements

Non-functional requirements arise through user needs because of budget constraints, organizational policies, the need for interoperability with other software or hardware systems, or external factors such as safety regulations or privacy legislation. Above figure is a classification of non-functional requirements. You can see from this diagram that the non-functional requirements may come from required characteristics of the software (product requirements), the organization developing the software (organizational requirements), or external sources:

1. **Product requirements:** These requirements specify or constrain the runtime behaviour of the software. Examples include performance requirements for how fast the system must execute and how much memory it requires; reliability requirements that set out the acceptable failure rate; security requirements; and usability requirements.

2. **Organizational requirements:** These requirements are broad system requirements derived from policies and procedures in the customer’s and developer’s organizations. Examples include operational process requirements that define how the system will be used; development process requirements that specify the programming language; the development environment or process standards to be used; and environmental requirements that specify the operating environment of the system.

3. **External requirements:** This broad heading covers all requirements that are derived from factors external to the system and its development

process. These may include regulatory requirements that set out what must be done for the system to be approved for use by a regulator, such as a nuclear safety authority; legislative requirements that must be followed to ensure that the system operates within the law; and ethical requirements that ensure that the system will be acceptable to its users and the general public.

3.3. REQUIREMENT ENGINEERING PROCESS

Requirements engineering involves three key activities. These are discovering requirements by interacting with stakeholders (elicitation and analysis); converting these requirements into a standard form (specification); and checking that the requirements actually define the system that the customer wants (validation). Requirements engineering is an iterative process in which the activities are interleaved.

The activities are organized as an iterative process around a spiral. The output of the RE process is a system requirements document. The amount of time and effort devoted to each activity in iteration depends on the stage of the overall process, the type of system being developed, and the budget that is available.

Early in the process, most effort will be spent on understanding high-level business and non-functional requirements, and the user requirements for the system. Later in the process, in the outer rings of the spiral, more effort will be devoted to eliciting and understanding the non-functional requirements and more detailed system requirements.

This spiral model accommodates approaches to development where the requirements are developed to different levels of detail. The number of iterations around the spiral can vary so that the spiral can be exited after some or all of the user requirements have been elicited. Agile development can be used instead of prototyping so that the requirements and the system implementation are developed together.

In virtually all systems, requirements change. The people involved develop a better understanding of what they want the software to do; the organization buying the system changes; and modifications are made to the system's hardware, software, and organizational environment. Changes have to be managed to understand the impact on other requirements and the cost and system implications of making the change.

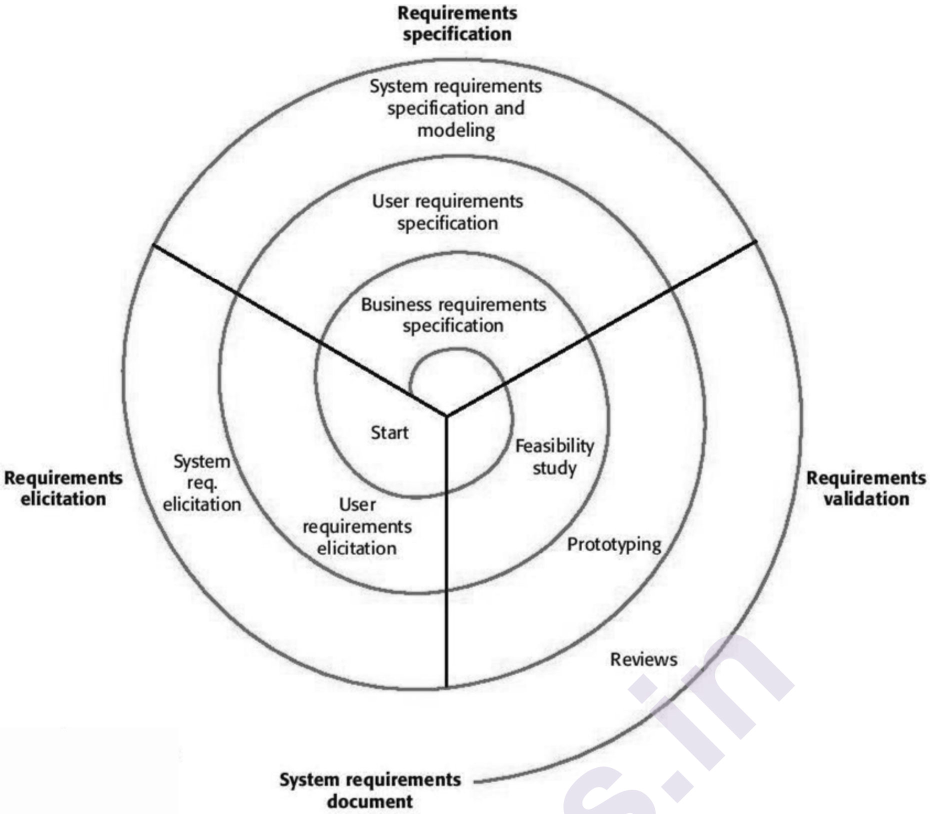

Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Prototyping
Reviews
System req. elicitation
User requirements elicitation
Requirements elicitation
Requirements validation
System requirements document

Figure: Spiral View

3.4. REQUIREMENT ANALYSIS AND DESIGN

Some of the problems that arise during the requirements engineering process are a result of failing to make a clear separation between these different levels of description. I distinguish between them by using the term user requirements to mean the high-level abstract requirements and system requirements to mean the detailed description of what the system should do.

User requirements and system requirements may be defined as follows:

1. User requirements are statements, in a natural language plus diagrams, of what services the system is expected to provide to system users and the constraints under which it must operate. The user requirements may vary from broad statements of the system features required to detailed, precise descriptions of the system functionality.

2. System requirements are more detailed descriptions of the software system’s functions, services, and operational constraints. The system requirements document (sometimes called a functional specification) should define exactly what is to be implemented. It may be part of the contract between the system buyer and the software developers.

Different kinds of requirement are needed to communicate information about a system to different types of reader. You need to write requirements at different levels of detail because different types of readers use them in different ways. The readers of the user requirements are not usually concerned with how the system will be implemented and may be managers who are not interested in the detailed facilities of the system. The readers of the system requirements need to know more precisely what the system will do because they are concerned with how it will support the business processes or because they are involved in the system implementation. System stakeholders include anyone who is affected by the system in some way and so anyone who has a legitimate interest in it. Stakeholders range from end-users of a system through managers to external stakeholders such as regulators who certify the acceptability of the system.

3.4.1 Data Flow Diagram

Introduction to data-flow diagrams

What are data-flow diagrams?

Data-flow diagrams (DFDs) model a perspective of the system that is most readily understood by users – The flow of information through the system and the activities that process this information.

Data-flow diagrams provide a graphical representation of the system that aims to be accessible to computer specialist and non-specialist users alike. The models enable software engineers, customers and users to work together effectively during the analysis and specification of requirements. Although this means that our customers are required to understand the modelling techniques and constructs, in data-flow modelling only a limited set of constructs are used, and the rules applied are designed to be simple and easy to follow. These same rules and constructs apply to all data-flow diagrams (i.e., for each of the different software process activities in which DFDs can be used).

The benefits of data-flow diagrams

Data-flow diagrams provide a very important tool for software engineering, for a number of reasons:

- The system scope and boundaries are clearly indicated on the diagrams (more will be described about the boundaries of systems and each DFD later in this chapter).

- The technique of decomposition of high level data-flow diagrams to a set of more detailed diagrams provides an overall view of the complete system, as well as a more detailed breakdown and description of individual activities, where this is appropriate, for clarification and understanding.

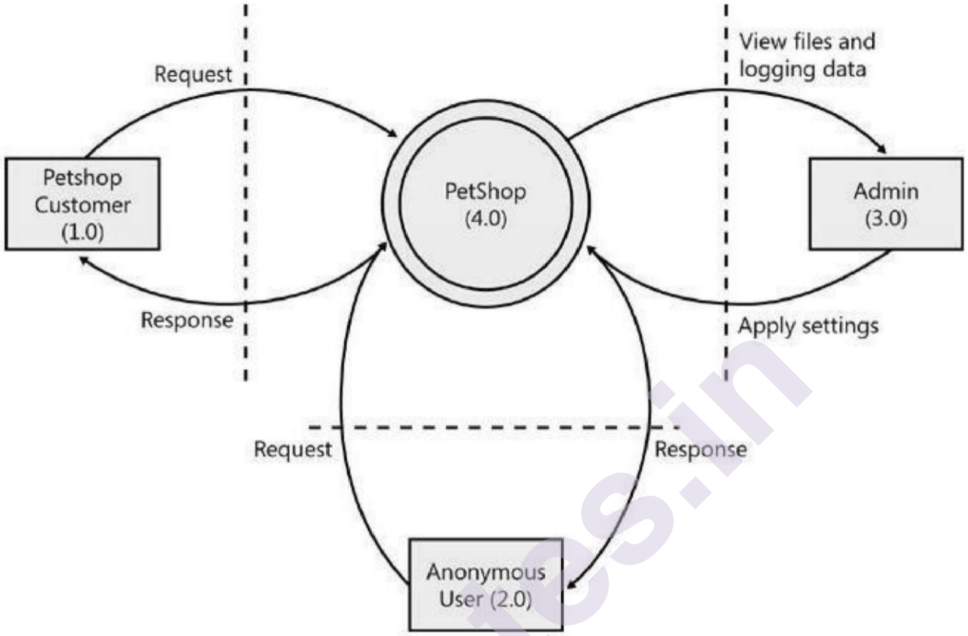
The different kinds (and levels) of data-flow diagrams:

Although all data-flow diagrams are composed of the same types of symbols, and the validation rules are the same for all DFDs, there are three main types of data-flow diagram:

Data-Flow Diagrams

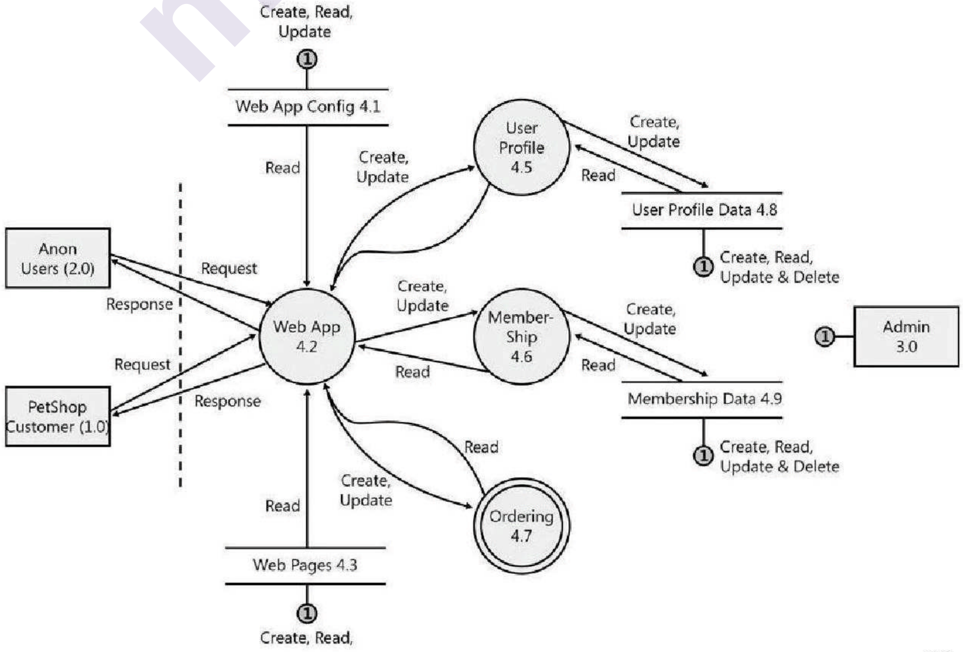
• **Context diagrams** — context diagram DFDs are diagrams that present an overview of the system and its interaction with the rest of the "world".

Sample DFD – Context



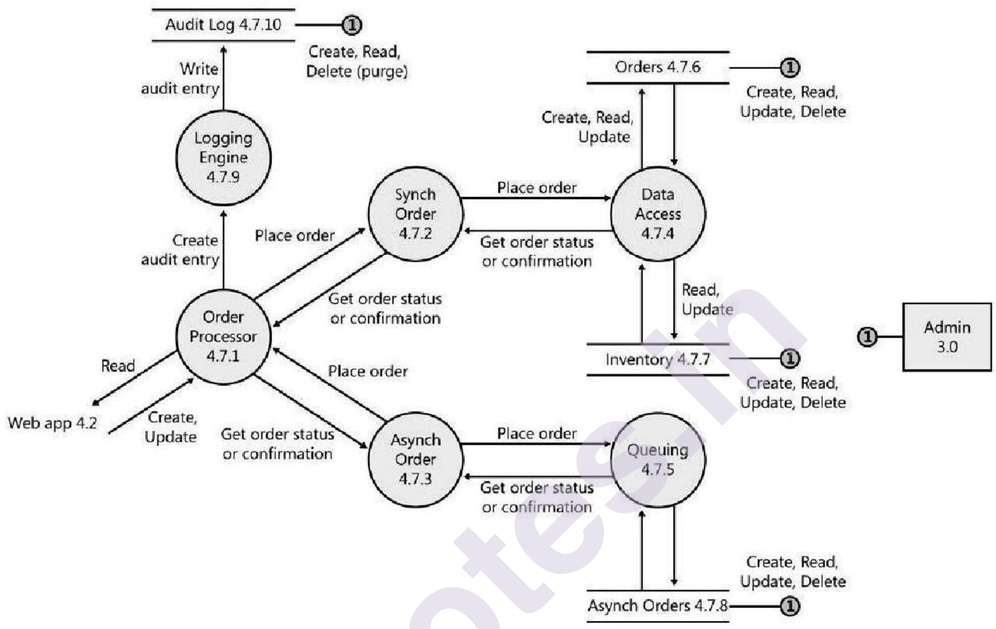
• **Level 1 data-flow diagrams** — Level 1 DFDs present a more detailed view of the system than context diagrams, by showing the main sub-processes and stores of data that make up the system as a whole.

Sample DFD – Level 1



- **Level 2 (and lower) data-flow diagrams** — a major advantage of the data-flow modelling technique is that, through a technique called "levelling", the detailed complexity of real world systems can be managed and modelled in a hierarchy of abstractions. Certain elements of any dataflow diagram can be decomposed ("exploded") into a more detailed model a level lower in the hierarchy.

Sample DFD – Level 2



During this unit we shall investigate each of the three types of diagram in the sequence they are described above. This is both a sequence of increasing complexity and sophistication, and also the sequence of DFDs that is usually followed when modelling systems.

For each type of diagram we shall first investigate what the features of the diagram are, and then we shall investigate how to create that type of diagram. However, before looking at particular kinds of dataflow diagrams, we shall briefly examine each of the symbols from which DFDs are composed.

3.4.2 Data Dictionary

There is a wide range of definitions for Data Dictionary /Directory Systems (DD/DS). Due to the increasing interest and the rapidly evolving nature of this field in recent years, terminology is somewhat confusing. One author speaks of a Data Dictionary System (DDS) or System Resources Dictionary, while another refers to DD/DS or Data Element Dictionary/Directory System, (DED/DS). Characteristic definitions include those of:

Leong-Hong and Marron (1977):

The DED/DS is a software tool that provides the means for defining and describing the characteristics of a database, as opposed to the contents of a database.

National Bureau of Standards (NBS) (1978):

The DED/DS is considered as a resource manager. It is an integrated repository that provides data necessary for managing data. Data management includes the planning, control, direction, and organization of data.

Allen, Loomis and Hannino (1982):

A DD/DS is an automated information system. It helps to achieve control of the data resource, by providing an inventory of that resource. It helps to control the cost of developing and maintaining applications. Finally it can provide for independence of metadata across comput¬ing environments, improving resiliency to the effects of hardware and software changes

Data has positive or negative value. The value of the data derives from the fact that the entire enterprise depends on its availability for the proper management of all other resources. Thus, it must be treated as a resource. The management and control of data resources begins with a proper definition and description of data. A DD/DS is a tool for the control and management of data as a resource.

To manage data as a resource it is basic that data about data must be clearly specified, easily accessible and well controlled. These data are called metadata. These are data objects, that in a data processing environment are represented in the form of elements, records, files or databases. Metadata is not user data, but identifies, defines and describes the characteristics of the latter. It describes the data resources of an organization. A DD/DS contains two types of metadata: Dictionary and Directory metadata. Dictionary metadata describes the data, and defines their meaning and structure. Directory metadata describes where the data is stored, and how internally represented and accessed.

A collection of related metadata comprises the metadata database. It consists of a database that contains descriptive and definitional information about the user database. It has basically the same characteristics of a user database. To achieve the goals of managing data as a resource it requires proper management. That is, planning for the design, implementation, maintenance, utilization and control. This implies that established lines of responsibility and authority; formal rules and detailed procedures to guide metadata- related activities; common procedures for collection, update, and maintenance; and common procedures for access control to the metadata must be developed. The DD/DS is the basic tool for managing the metadata database.

The DD/DS is divided into three categories, based on the scope of control exercised through metadata management: Active, Potentially Active and Passive. A DD/DS is said to be active with respect to a program or process if that program or process is fully dependent on the DD/DS for its metadata. A DD/DS is said to be passive if it does not generate metadata and does not have control over where and how a user or processing component obtains the required metadata. A potentially active DD/DS provides the capability of producing the metadata for a given program or process. A potentially active DD/DS can be extended to active through supportive administrative procedures. Many of the currently commercially available DD/DSs are of this type. In practice, the concept of active/passive DD/DS refers to interfaces that it provides to other software packages. A DD/DS with active interfaces can better serve the goals of managing data as a resource.

3.4.3 HIPO Chart

Hierarchy-Input-Process-Output (HIPO)

Hierarchy-Input-Process-Output (HIPO) is a documentation technique developed by IBM and used in a number of different situations. It is useful for design documentation as well as stating requirements and specifications before beginning design. It is based on two concepts: The input-process-output model of information processing and functional hierarchies. HIPO is basically a graphical notation, consisting of hierarchy charts and process charts. The hierarchy chart is similar to a structure chart. Each box in the chart can represent a system, subsystem, program, or program module. Its purpose is to show the overall functional components and to refer to overview and detail HIPO diagrams. It does not show the data flow between functional components or any control information. It does not show the arrows with open or filled circles. Also, it does not give any information about the data components of the system or program.

The second part of the HIPO notation is process charts. Process charts have two levels: overview diagram and detail diagrams. They are used to describe a function in terms of its inputs, the processing to be done, and the outputs produced. Any of the information, especially the process part, may be described in more detail by accompanying textual material. The process charts show the flow of data through processes; however, they are more difficult to draw than data flow diagrams. HIPO diagrams often require more verbiage and symbols to give the same information as a comparable data flow diagram. HIPO has been used in a number of DP (Data Processing) situations and some more complex specification tasks. Its drawbacks, especially with respect to interfaces and the description of data, were limited to its acceptance. HIPO does not include any guidelines, strategies, or procedures to guide the analyst in building a functional specification or the designer in building a system or program design.

At the general level, a structure chart is usually preferred over a hierarchy chart. At the detailed level, pseudocode is often preferred over HIPO diagrams because it provides more information in a more compact form. HIPO diagrams have no symbols for representing detailed program structures such as conditions, case structures, and loops. HIPO diagrams cannot represent data structures or the linkage to data models.

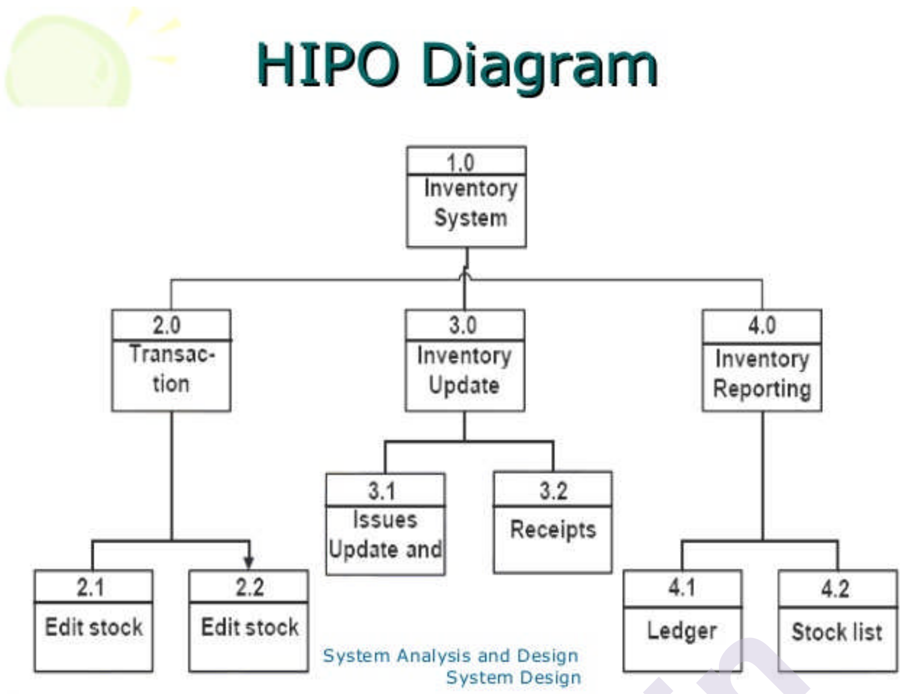


Figure: HIPO Chart

3.4.4 Warnier Orr Diagram

A graphical representation of a horizontal hierarchy with brackets isolating the levels is utilized to plan or document a data structure, a set of detailed logic, a program, or a system.

Strengths, weaknesses, and limitations:

Warnier-Orr diagrams are excellent tools for describing, planning, or documenting data structures. They can show a data structure or a logical structure at a glance. Because only a limited number of symbols are required, specialized software is unnecessary and diagrams can be created quickly by hand. The basic elements of the technique are easy to learn and easy to explain. Warnier-Orr diagrams map well to structured code.

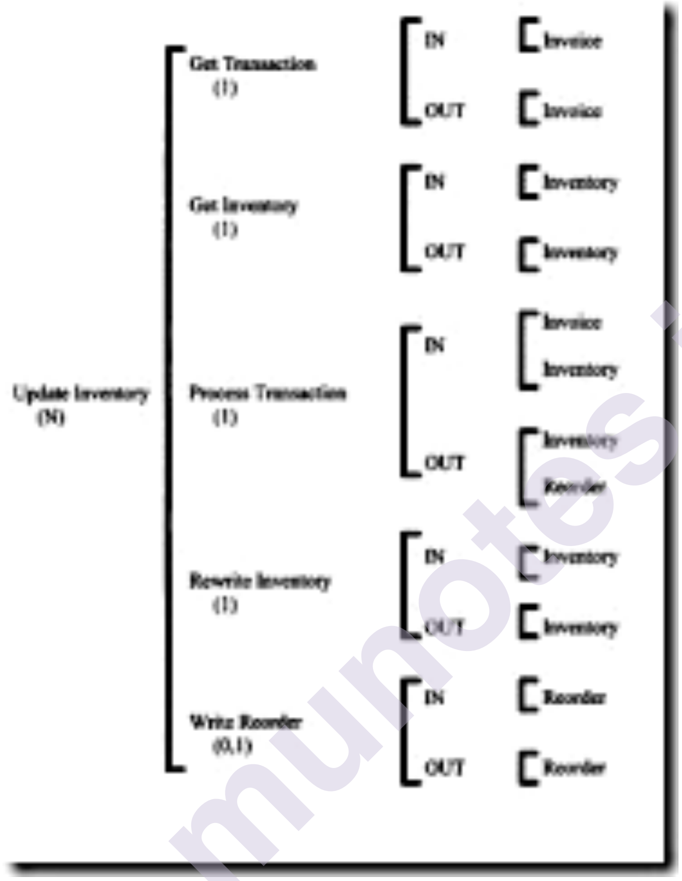
Concepts:

The Warnier-Orr design methodology, also known as the structured requirements definition methodology was developed in the early 1970s by Warnier and extended to system design by Orr. The first step in the methodology is to create entity diagrams for each major user. The entity diagrams are then merged to create a system entity diagram, and the major tasks that must be performed are derived from the system's data requirements.

In-out diagrams:

A Warnier-Orr diagram shows a data structure or a logical structure as a horizontal hierarchy with brackets separating the levels. Once the major tasks are identified, the systems analyst or information system consultant prepares an in-out Warnier-Orr diagram to document the application's primary inputs and outputs.

For example, Figure shows an in-out diagram for a batch inventory update application. Start at the left (the top of the hierarchy). The large bracket shows that the program, Update Inventory, performs five primary processes ranging from Get Transaction at the top to Write Reorder at the bottom. The letter N in parentheses under Update Inventory means that the program is repeated many (1 or more) times. The digit 1 in parentheses under Get Transaction (and the next three processes) means the process is performed once. The (0, 1) under Write Reorder means the process is repeated 0 or 1 times, depending on a run-time condition. (Stock may or may not be reordered as a result of any given transaction.)



Data flow into and out from every process. The process inputs and outputs are identified to the right of the in-out diagram. For example, the Get Transaction process reads an Invoice and passes it to a subsequent process. The last column is a list of the program's primary input and output data structures. Note how the brackets indicate the hierarchical levels.

3.5 REQUIREMENT ELICITATION

The aims of the requirements elicitation process are to understand the work that Stakeholders do and how they might use a new system to help support that work.

During requirements elicitation, software engineers work with stakeholders to find out about the application domain, work activities, the services and system features that stakeholders want, the required performance of the system, hardware constraints, and so on. Eliciting and understanding requirements from system stakeholders is a difficult process for several reasons:

1. Stakeholders often don't know what they want from a computer system except in the most general terms; they may find it difficult to articulate what they want the system to do; they may make unrealistic demands because they don't know what is and isn't feasible.

2. Stakeholders in a system naturally express requirements in their own terms and with implicit knowledge of their own work. Requirements engineers, without experience in the customer's domain, may not understand these requirements.

3. Different stakeholders, with diverse requirements, may express their requirements in different ways. Requirements engineers have to discover all potential sources of requirements and discover commonalities and conflict.

4. Political factors may influence the requirements of a system. Managers may demand specific system requirements because these will allow them to increase their influence in the organization.

5. The economic and business environment in which the analysis takes place is dynamic. It inevitably changes during the analysis process. The importance of particular requirements may change. New requirements may emerge from new stakeholders who were not originally consulted.

3.5.1 Interviewing

Formal or informal interviews with system stakeholders are part of most requirements Engineering processes. In these interviews, the requirements engineering team puts questions to stakeholders about the system that they currently use and the system to be developed. Requirements are derived from the answers to these questions.

Interviews may be of two types:

1. Closed interviews, where the stakeholder answers a predefined set of questions.

2. Open interviews, in which there is no predefined agenda. The requirements engineering team explores a range of issues with system stakeholders and hence develops a better understanding of their needs.

In practice, interviews with stakeholders are normally a mixture of both of these. You may have to obtain the answer to certain questions, but these usually lead to other issues that are discussed in a less structured way. Completely open-ended discussions rarely work well. You usually have to ask some questions to get started and to keep the interview focused on the system to be developed.

Interviews are good for getting an overall understanding of what stakeholders do, how they might interact with the new system, and the difficulties that they face with current systems. People like talking about their work, and so they are usually happy to get involved in interviews. However, unless you have a system prototype to demonstrate, you should not expect stakeholders to suggest specific and detailed requirements. Everyone finds it difficult to visualize what a system might be like. You need to analyze the information collected and to generate the requirements from this.

Eliciting domain knowledge through interviews can be difficult, for two reasons:

1. All application specialists use jargon specific to their area of work. It is impossible for them to discuss domain requirements without using this terminology. They normally use words in a precise and subtle way that requirements engineers may misunderstand.

2. Some domain knowledge is so familiar to stakeholders that they either find it difficult to explain or they think it is so fundamental that it isn't worth mentioning. For example, for a librarian, it goes without saying that all acquisitions are catalogued before they are added to the library. However, this may not be obvious to the interviewer, and so it isn't taken into account in the requirements.

Interviews are not an effective technique for eliciting knowledge about organizational requirements and constraints because there are subtle power relationships between the different people in the organization. Published organizational structures rarely match the reality of decision making in an organization, but interviewees may not wish to reveal the actual rather than the theoretical structure to a stranger. In general, most people are generally reluctant to discuss political and organizational issues that may affect the requirements.

To be an effective interviewer, you should bear two things in mind:

1. You should be open-minded, avoid preconceived ideas about the requirements, and willing to listen to stakeholders. If the stakeholder comes up with surprising requirements, then you should be willing to change your mind about the system.

2. You should prompt the interviewee to get discussions going by using a springboard question or a requirements proposal or by working together on a prototype system. Saying to people "tell me what you want" is unlikely to result in useful information. They find it much easier to talk in a defined context rather than in general terms.

Information from interviews is used along with other information about the system from documentation describing business processes or existing systems, user observations, and developer experience. Sometimes, apart from the information in the system documents, the interview information may be the only source of information about the system requirements. However, interviewing on its own is liable to miss essential information, and so it should be used in conjunction with other requirements elicitation techniques.

3.5.2 Questionnaire

Interviews – Start Up Questions

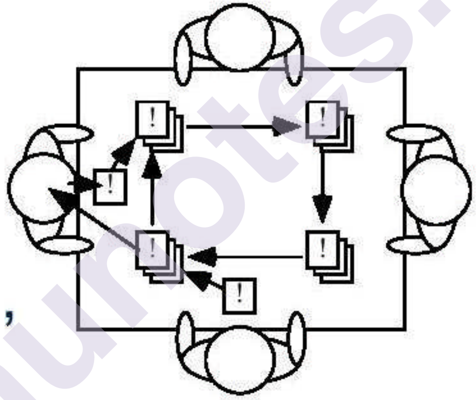
- Context-free questions to narrow the scope a bit (Weinberg)
- Identify customers, goals, and benefits
- Who is (really) behind the request for the system?
- Who will use the system? Willingly?
- Are there several types of users?
- What is the potential economic benefit from a successful solution?
- Is a (pre-existing) solution available from another source?
- When do you need it by?
- Can you prioritize your needs?
- What are your constraints?
- Time
- Budget
- Resources (human or otherwise)
- Expected milestones (deliverables and dates)?
- Try to characterize the problem and its solution
- What would be a "good" solution to the problem?
- What problems is the system trying to address?
- In what environment will the system be used?
- Any special performance issues?
- Other special constraints?
- What is (un)likely to change?
- Future evolution?
- What needs to be flexible (vs. quick & dirty)?
- Questions that cannot be asked directly (ask indirectly)
- Are you opposed to the system?
- Are you trying to obstruct/delay the system?
- Are you trying to create a more important role for yourself?
- Do you feel threatened by the proposed system?
- Are you trying to protect your job?
- Is your job threatened by the new system?
- Is anyone else's?

3.5.3 Brainstorming

- To invent new way of doing things or when much is unknown
- When there are few or too many ideas
- Early on in a project particularly when:

- Terrain is uncertain
- There is little expertise for the type of applications
- Innovation is important (e.g., novel system)

- Two main activities:
- The Storm: Generating as many ideas as possible (quantity, not quality) – wild is good!
- The Calm: Filtering out of ideas (combine, clarify, prioritize, improve…) to keep the best one(s) – may require some voting strategy



- Roles: scribe, moderator (may also provoke), Participants

Brainstorming – Objectives

- Hear ideas from everyone, especially unconventional ideas
- Keep the tone informal and non-judgemental
- Keep the number of participants "reasonable" – if too many, consider a "playoff "-type filtering and invite back the most creative to multiple sessions
- Encourage creativity
- Choose good, provocative project name.
- Choose good, provocative problem statement
- Get a room without distractions, but with good acoustics, whiteboards, coloured pens, provide coffee/donuts/pizza/beer
- Provide appropriate props/mock-ups

Brainstorming – Roles

- Scribe
- Write down all ideas (may also contribute)
- May ask clarifying questions during first phase but without criticizing
- Moderator/Leader
- Cannot be the scribe
- Two schools of thought: traffic cop or agent provocateur
- Traffic cop – enforces "rules of order", but does not throw his/her weight around otherwise
- Agent provocateur – traffic cop plus more of a leadership role, comes prepared with wild ideas and throws them out as discussion wanes
- May also explicitly look for variations and combinations of other suggestions

Brainstorming – Participants

- Virtually any stakeholder, e.g.
- Developers
- Domain experts
- End-users
- Clients
- ...
- “Ideas-people” – a company may have a special team of people
- Chair or participate in brainstorming sessions
- Not necessarily further involved with the project

Brainstorming – The Storm

- Goal is to generate as many ideas as possible
- Quantity, not quality, is the goal at this stage
- Look to combine or vary ideas already suggested
- No criticism or debate is permitted – do not want to inhibit participants
- Participants understand nothing they say will be held against them later on
- Scribe writes down all ideas where everyone can see
- E.g., whiteboard, paper taped to wall
- Ideas do not leave the room
- Wild is good
- Feel free to be gloriously wrong
- Participants should NOT censor themselves or take too long to consider whether an idea is practical or not – let you go!

Brainstorming – The Calm

- Go over the list of ideas and explain them more clearly
- Categorize into "maybe" and "no" by pre-agreed consensus method

- Informal consensus
- 50% + 1 vote vs. "clear majority"
- Does anyone have veto power?
- Be careful about time and people
- Meetings (especially if creative or technical in nature) tend to lose focus after 90 to 120 minutes – take breaks or reconvene later
- Be careful not to offend participants
- Review, consolidate, combine, clarify and improve.
- Rank the list by priority somehow
- Choose the winning idea(s)

Brainstorming – Tool Support

- With many good ideas, some outrageous and even farfetched, brainstorming can be really fun!
- Creates a great environment that stimulates people and motivates them to perform well!
- Can be done by email, but a good moderator/leader is needed to
- Prevent flamers to come into play
- Prevent race conditions due to the asynchronous communication medium
- Be careful not to go into too much detail
- Collaboration tools are also possible
- TWiki and many other more appropriate tools such as BrainStorm and IdeaFisher

3.5.4 Facilitated Application Specification Technique

Its objective is to bridge the expectation gap – difference between what the developers think they are supposed to build and what customers think they are going to get.

A team oriented approach is developed for requirements gathering. Each attendee is asked to make a list of objects that are-

1. Part of the environment that surrounds the system
2. Produced by the system
3. Used by the system

Each participant prepares his/her list, different lists are then combined, redundant entries are eliminated, team is divided into smaller sub-teams to develop mini-specifications and finally a draft of specifications is written down using all the inputs from the meeting.

3.5.5 Use Case Approach

Use cases are a way of describing interactions between users and a system using a graphical model and structured text. They were first introduced in the Objectory method (Jacobsen et al. 1993) and have now become a fundamental feature of the Unified Modelling Language (UML). In their simplest form, a use case identifies the actors involved in an interaction and names the type of interaction. You then add additional information describing the interaction with the system. The additional information may be a textual description or one or more graphical models such as the UML sequence or state charts (see Chapter 5). Use cases are documented using a high-level use case diagram. The set of use cases represents all of the possible interactions that will be described in the system requirements. Actors in the process, who may be human or other systems, are represented as stick figures. Each class of interaction is represented as a named ellipse. Lines link the actors with the interaction. Optionally, arrowheads may be added to lines to show how the interaction is initiated. This is illustrated in Figure 4.15, which shows some of the use cases for the Mentcare system. Use cases identify the individual interactions between the system and its users or other systems. Each use case should be documented with a textual description. These can then be linked to other models in the UML that will develop the scenario in more detail. For example, a brief description of the Setup Consultation use case from Figure.

Setup consultation allows two or more doctors, working in different offices, to view the same patient record at the same time. One doctor initiates the consultation by choosing the people involved from a dropdown menu of doctors who are online. The patient record is then displayed on their screens, but only the initiating doctor can edit the record. In addition, a text chat window is created to help coordinate actions. It is assumed that a phone call for voice communication can be separately arranged.

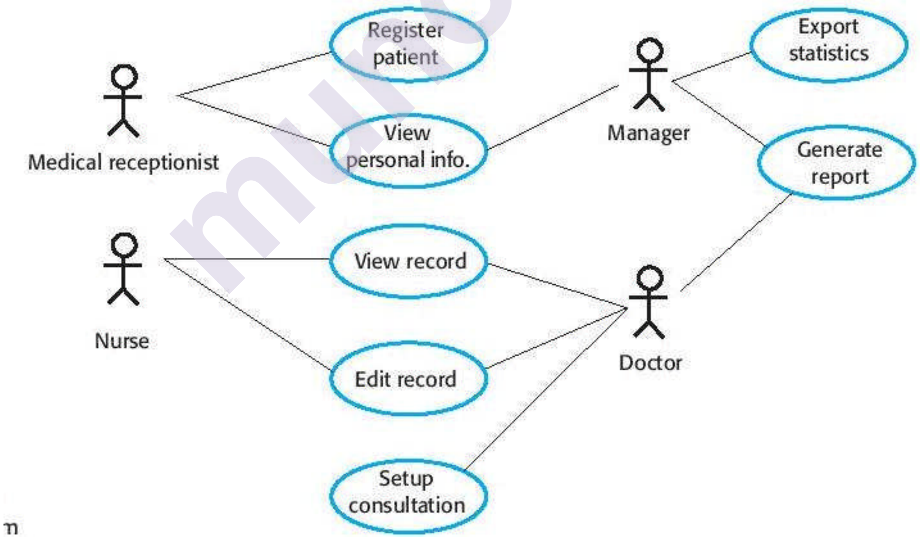


Figure: Use Case Diagram

The UML is a standard for object-oriented modeling, so use cases and use case based elicitation are used in the requirements engineering process. However, my experience with use cases is that they are too fine-grained to be useful in discussing requirements. Stakeholders don’t understand the term use case; they don’t find the graphical model to be useful, and they are often not interested in a detailed description of each and every system interaction. Consequently, I find use cases to be more helpful in systems design than in requirements engineering. I discuss use cases further in Chapter 5, which shows how they are used alongside other system models to document a system design.

Some people think that each use case is a single, low-level interaction scenario. Others, such as Stevens and Pooley (Stevens and Pooley 2006), suggest that each use case includes a set of related, low-level scenarios. Each of these scenarios is a single thread through the use case. Therefore, there would be a scenario for the normal interaction plus scenarios for each possible exception. In practice, you can use them in either way.

3.6 SUMMARY

- Requirements for a software system set out what the system should do and define constraints on its operation and implementation

- Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out.

- Non-functional requirements often constrain the system being developed and the development process being used. These might be product requirements, organizational requirements, or external requirements. They often relate to the emergent properties of the system and therefore apply to the system as a whole.

- The requirements engineering process includes requirements elicitation, requirements specification, requirements validation, and requirements management

- Requirements elicitation is an iterative process that can be represented as a spiral of activities requirements discovery, requirements classification and organization, requirements negotiation, and requirements documentation.

Reference Books:

1. Software Engineering 6th edition Roger S. Pressman
2. Software Engineering Tenth Edition Ian Sommerville, Pearson.

❖❖❖❖

4

OBJECT ORIENTED ANALYSIS AND DESIGN

Unit Structure

4.0 OBJECTIVE

After studying this particular chapter, you will be able to understand

- To know what is UML
- To know the features of UML
- To know what is UML not
- To know the Object Oriented Concepts
- To understand need of diagrams and different model element used in it.

4.1 INTRODUCTION

UML stands for Unified Modeling Language. It is basically visual modeling language that helps to create blueprints that capture vision in standard, understandable form.

The UML is nothing but the graphical language which explains the following artifacts of software system.

1. Specifying- Define model precisely, unambiguously and completely
2. Visualizing- Define Graphical Notations to communicate with system ambiguously.
3. Constructing – Define design dimensions for building models.
4. Documenting – Define documentation for inception.

Models are representation of reality. Software under development can be complex so models of software can be used. One such modeling technique is **Unified Modeling Language.**

Models are created before actual development of the systems so that we can understand possibility, problems, options etc. The different views give different perspective of the system when models are created.

Example- car driver is interested in dashboard and its functions, but if we consider electronics person he shows his interest in circuit connections.

4.1.1 Review of object orientation

The UML can be used in better way with the Object Oriented Technology. So now we will see these concepts

1.	Object	• The things that can be tangible. • Instance of a class • Things has some kind of identity, state, behavior Example- A car(brand name ,color ,type etc.)
2.	Class	• Group of structurally identical items • Encapsulation of data and operations. Example- A Car Attribute-color, brand name etc Operations – start(), stop(),accelerate() etc.
3.	Encapsulation	• Wrapping of data and function in single unit. • The data is not directly accessible to outside world Example- for car class operations are around the attributes.
4.	Abstraction	• Representing essential features without including background explanation. Example- A car gives information about attributes and operations not its process.

5.	Inheritance	• A process by which object of one class acquire the properties of other class. • It is known as super class and sub class or parent class and child class. Example- vehicle is super class and car, auto, truck are subclass
6	Polymorphism	• Ability to take more than one form Example – All polygon occupies area. Triangle ,rectangle are polygon.
7.	Message Passing	• Communication between set of objects Example- AC operated by remote (on, off, temp increased or decrease). Here object remote invoke the function of object AC.

4.2 UML OVERVIEW

UML is constructed from three building blocks: -Things, relationship, diagram.

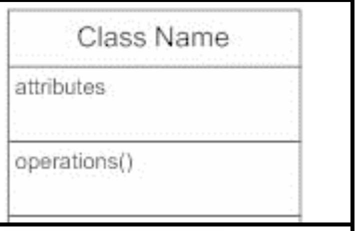
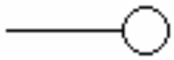
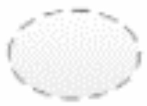
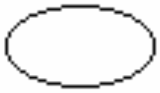
To use UML, it is necessary to understand these building block clearly.

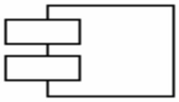
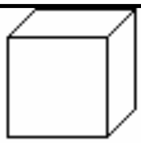
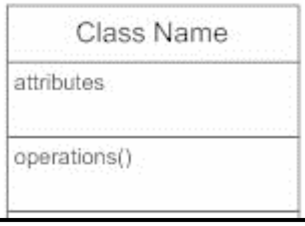
4.2.1.1 Things

4.2.1.1.1 Structural Things –

- It is also called as nouns of the UML model.
- It is mostly representing the static part of the system.
- The conceptual or physical elements are represented by structural things.

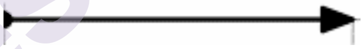
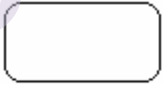
Some structural things which are available with UML-

Sr no	Structural Things	Symbols
1	Class- Abstact set of things contain similar properties and functions Example- Car class	
2	Interface – Set of operation which tells clearly what a class can do. Example- car class, bus class contain common action start engine()	
3	Collaboration- Provide an interaction between things so that goal have achieved. Example- Reservation System	
4	Use case- Set of action that system performed to achieve specific goal. Example- Admission System	

5	Component – Describe physical part of system. Example- Set of classes.	
6	Node – Run time physical element. Example-	
7	Active classes – It is a class whose objects having one or more processes	

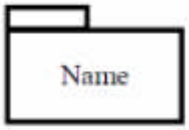
4.2.1.1.2 Behavioral Things-

- It is basically depending upon behavior (called dynamic also).
- It is also called verbs of UML.
- In behavioral things behaviors are represented based on time and space.

Sr no	Behavioral Things	Symbols
1.	Interaction – Exchange of message between objects.	
2.	State Machine- Sequences of state of objects used in its lifetime to give response to event. Example-ATM System	

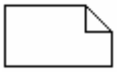
4.2.1.1.3 Grouping things

- Grouping refers to organizational part of UML model.
- Higher level abstraction is provided by grouping things.
- In grouping model can be decomposed.

Sr no	Grouping Things	Symbols
1.	Package – It exist only during development time. It includes structural, behavioral and other grouping things	

4.2.1.1.4 Annotation things

- If we wish to add comments to UML diagrams for more explanation purpose, then annotation things are used.

Sr.no	Annotation Things	Symbols
1.	Note- Graphical notation for attaching comments, rules etc.	

Relationship

- Relationship is nothing but the links between the things which shows how system is implemented.
- There are four types of relationships

1. **Dependency-**

- In this relationship change in one affects the other.
- The notation used for this is - - - - - - ->
- Example- If date of birth of a person changed then age will automatically Change.

2. **Association-**

- It is structural relationship.
- It shows how objects are taking part in relationship.
- The notation used for this ————————
- Example- Relationship between doctor and patient.

3. **Generalization-**

- It is a relationship between Parent class and child class.
- The notation used for this ◁————————
- Example- Account class contain two sub class Saving account and current Account

4. Realization-

- It is semantic relationship.
- In this on specifies behavior to be carried and othe carries out behavior.
- The notation used for this - - - - - - - - -▷
- Example- AC carries out the behavior specified by remote.

4.2.1.2 Diagrams

- The diagrams are graphical representation of models which uses various symbols and text.
- There are Following types of UML Diagrams-

1. Structure diagrams-

- Set of diagrams that mention the elements of specification.
- It represents framework for system.
- It includes class, structure, component, deployment diagrams.

2. Behavior diagram-

- It represents features of system.
- It describes the interaction in the system.
- It includes activity, use-case, state machine and interaction diagram.

3. Interaction Diagram-

- This type comes under behavior diagram.
- It emphasizes object interaction.
- It includes communication, interaction and sequence diagram.

4.3 NATURE AND PURPOSE OF UML MODEL

Nature of UML Model-

- It is a representation in a certain medium of something in the same or another medium
- It captures the important aspects of the thing being.
- Model of a software system is made in a modelling language, such as UML.
- It generator of potential configurations of systems; the possible systems are its extent, or values.
- Model has both semantics and notation and can take various forms that include both pictures and text.
- The levels of models give guidance to thought process and also gives abstract specification of the system.
- The models can be

1. Conceptual models

✓ describe a situation of interest in the world, such as a business operation or factory process.

✓ Nothing about how much of the model is implemented or supported by software.

2. Specification models

✓ define what a software system must do, the information it must hold, and the behaviour it must exhibit.

✓ They assume an ideal computing platform.

3. Implementation models

✓ describe how the software is implemented, considering all the computing environment's constraints and limitations.

Purpose of UML Model-

- To capture and precisely state requirements and domain knowledge so that all capture requirements about the appearance, traffic patterns, various kinds of utility services, strength against wind and earthquakes, cost, and many other things.

✓ Different models of a software system may capture requirements about its application domain, the ways users will use it, its breakdown into modules, common patterns used in its construction, and other things.

- To think about the design of a system.

✓ software model system helps developers explore several architectures and design solutions easily before writing code.

✓ good modelling language allows the designer to get the overall architecture right before detailed design begins.

- To capture design decisions in a mutable form separate from the requirements.

✓ One model of a software system can capture the external behaviour of a system and the real-world domain information represented by the system.

✓ Another model shows the internal classes and operations that implement the external behaviour.

- To generate usable work products.

✓ model of a software system can be used to generate class declarations, procedure bodies, user interfaces, databases, scenarios of legal use, configuration scripts, and lists of race conditions.

- To organize, find, filter, retrieve, examine, and edit information about large systems.

✓ model of a software system organizes information into several views: static structure, state machines, interactions, requirements, and so on.

- To explore multiple solutions economically.

✓ Models of a large software system permit several designs to be proposed and compared.

✓ The models are not constructed in full detail, of course, but even a rough model can expose many issues the final design must deal with.

- To master complex systems.

✓ A model of a large software system permits dealing with complexity that is too difficult to deal with directly.

✓ A model can determine the potential impact of a change before it is made, by exploring dependencies in the system.

4.4 FEATURES OF UML

UML provides following features-

1. Syntax only-

UML is a language not a notation so it used to describe that which model elements are used with diagrams and how to used it.

2. Comprehensive-

It can be used to model anything even though the modeling syntax aimed primarily at creating models of software based system.

3. 13 diagrams-

UML includes total 13 diagrams which helps in different perspectives of the system. In practice we are not using all of them, we used them as per requirements.

4. Language-independent-

It has no binding to any type of language. The binding can be decided based on which tool we select for representation.

5. Process independent-

Any process can be used for creating models. It has no binding with the process for creating models.

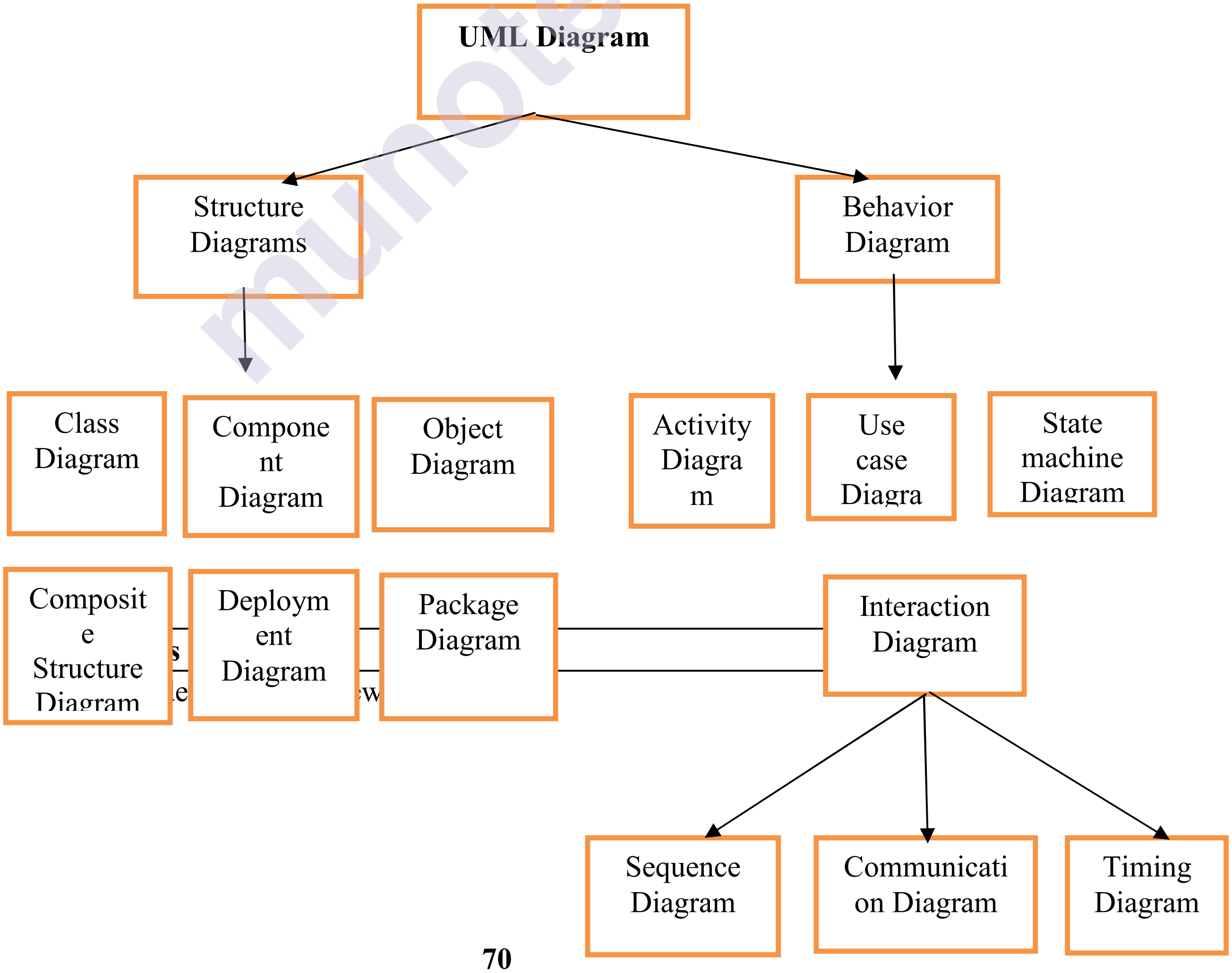
6. Tool-independent-

There are restrictions on the vendors who provide the tools for drawing diagrams**.** Each vendor can do value addition to the visual modelling of diagrams.

7. Well documented-

The UML diagrams having well documentation which describe the notations and its examples.

4.5 UML DIAGRAMS



- It is also known as structural diagram as it shows the collection of classes, interfaces, association, collaboration.
- Class diagram shows the different classes involved and the relationship between classes.
- It describes the functionality shown by the system.

Purpose of class Diagram-

- Analysis and designing of static view of application
- To describe responsibility of system
- Base for drawing component and deployment diagrams.
- Creating an application with the help of given requirements.

How to draw class diagram?

The following points should be remembered while drawing a class diagram –

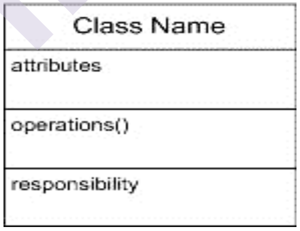
1. Class name should be meaningful.
2. Each element of the class with its relationship should be identified earlier
3. Attributes and methods of a class should be clearly mentioned
4. For each class minimum number of properties should be mentioned clearly.
5. To understand the diagram properly add notes in it.
6. Last but not the least prepare the diagram on paper to make it final.

Basic Class Diagram Symbols and Notations

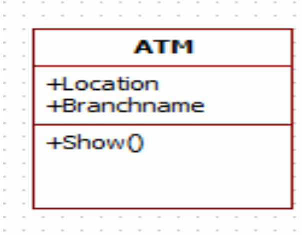
1. Class –

- It represents abstraction of entity having common characteristics
- Class denote with rectangle symbol divided into compartments.
- Name of the class in first compartment (centred, bolded, and capitalized), list of attributes in second compartment (left-aligned, not bolded, and lowercase), write operations into the third compartment.

Symbol



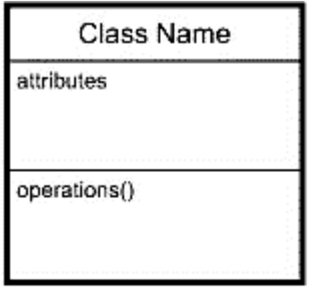
Example



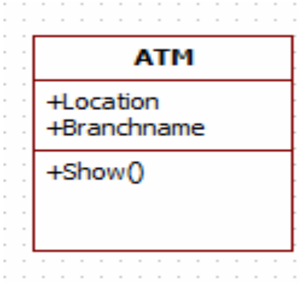
2. Active classes-

- Active classes initiate and control the flow of activity
- Active classes shown with a thicker border.

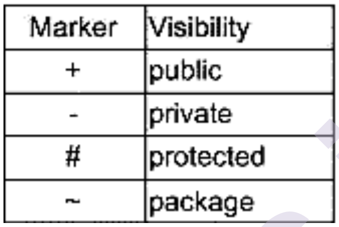
Symbol



Example



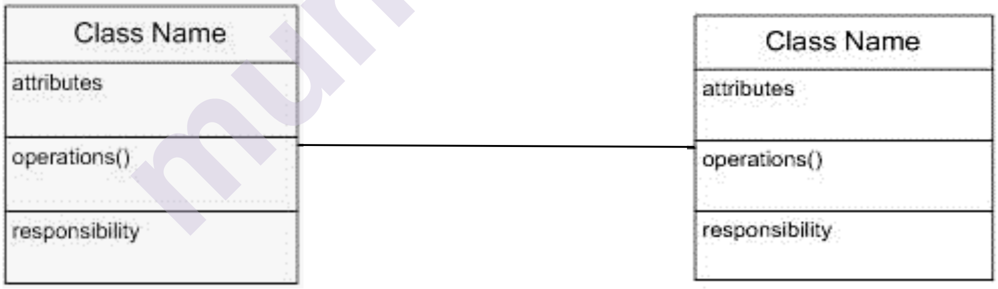
3. Visibility
- It is used to describe that who can access the information which is present in class.
- Symbols for the visibility are as follows-



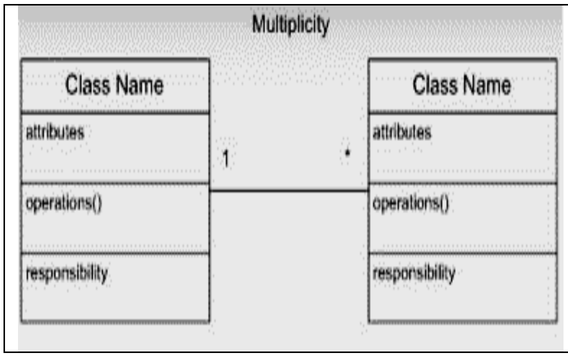
Marker	Visibility
+	public
-	private
#	protected
~	package

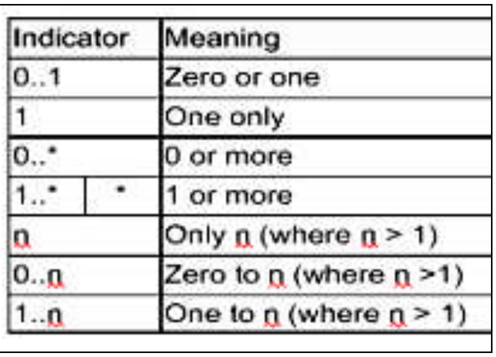
Public-Allow all other classes to view information
Private- Hide information from anything outside the class.
Protected- Allows child class to access information from parent class.

4. Association
- It is used to represent static relationship between classes.
- It is going to be shown by solid line



5. Multiplicity
- This is used to show the relationship between two classes.
- This notation is placed near the end of association line.
- The symbols are shown below

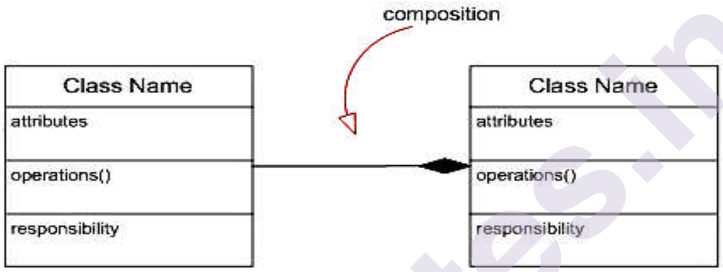




Indicator	Meaning
0..1	Zero or one
1	One only
0..*	0 or more
1..* *	1 or more
n	Only n (where n > 1)
0..n	Zero to n (where n >1)
1..n	One to n (where n > 1)

6. Composition

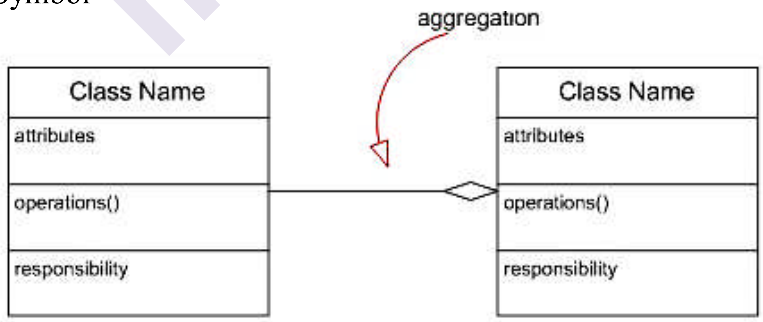
- ➢ Composition is special type of aggregation.
- ➢ It shows strong relationship between classes.
- ➢ In this one class is whole and other is part.
- ➢ Composition shown by line with filled diamond at the end.
- ➢ Symbol



- ➢ Example- ROOM class does not exist separate to HOUSE class.

7. Aggregation-

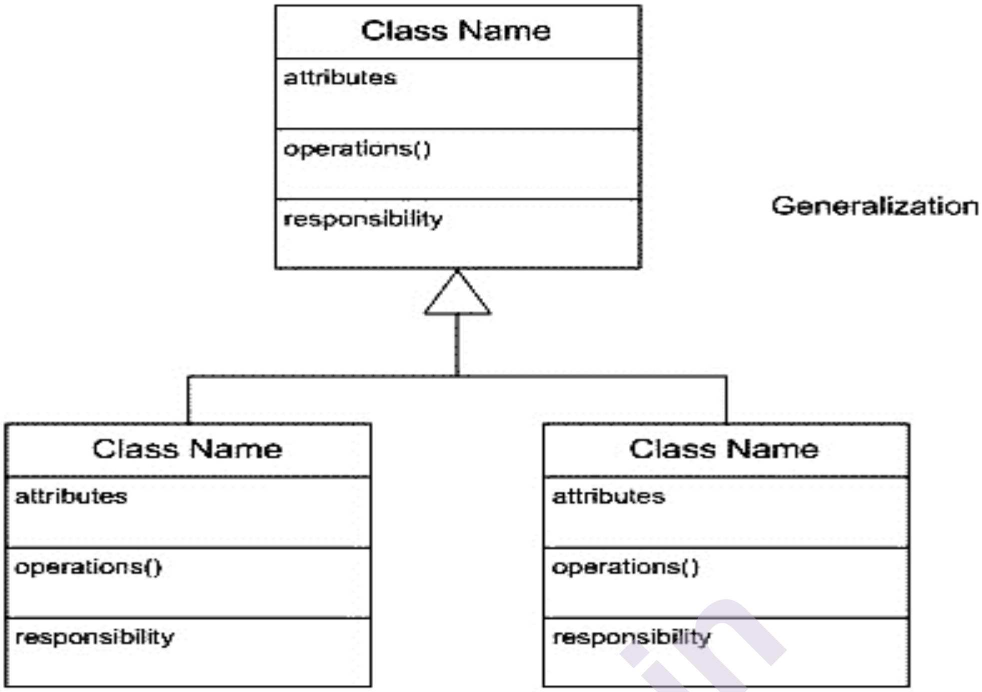
- ➢ In this whole class plays more important role than part class.
- ➢ Classes are not depending upon each other.
- ➢ Aggregation shown by hollow diamond.
- ➢ Symbol



➢ Example-Class DEPARTMENT having association between class COMPANY. Department is part of company.

8. Generalization-

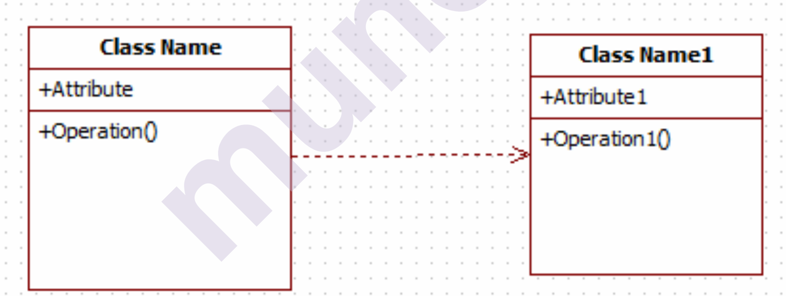
- ➢ It is nothing but an inheritance.
- ➢ It is showing relationship between parent and child class.
- ➢ It indicates “ is a” relationship
- ➢ Symbol



- Example – MANGO class, ORANGE class are part of FRUIT Class.

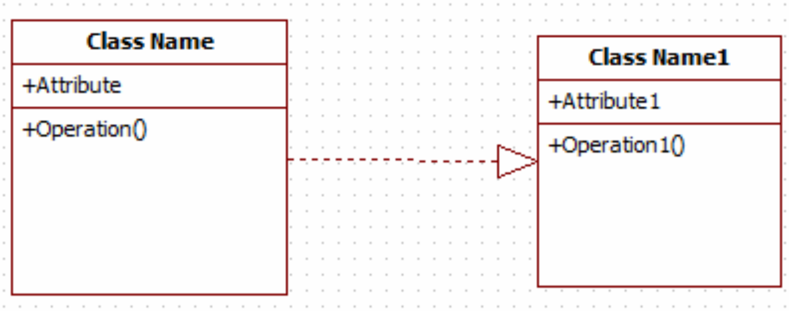
9. Dependence-

- It is special type of association.
- If any changes happen in one class, it affects the changes in other class.
- It is shown by dashed line with normal arrow head.
- Symbol shown as

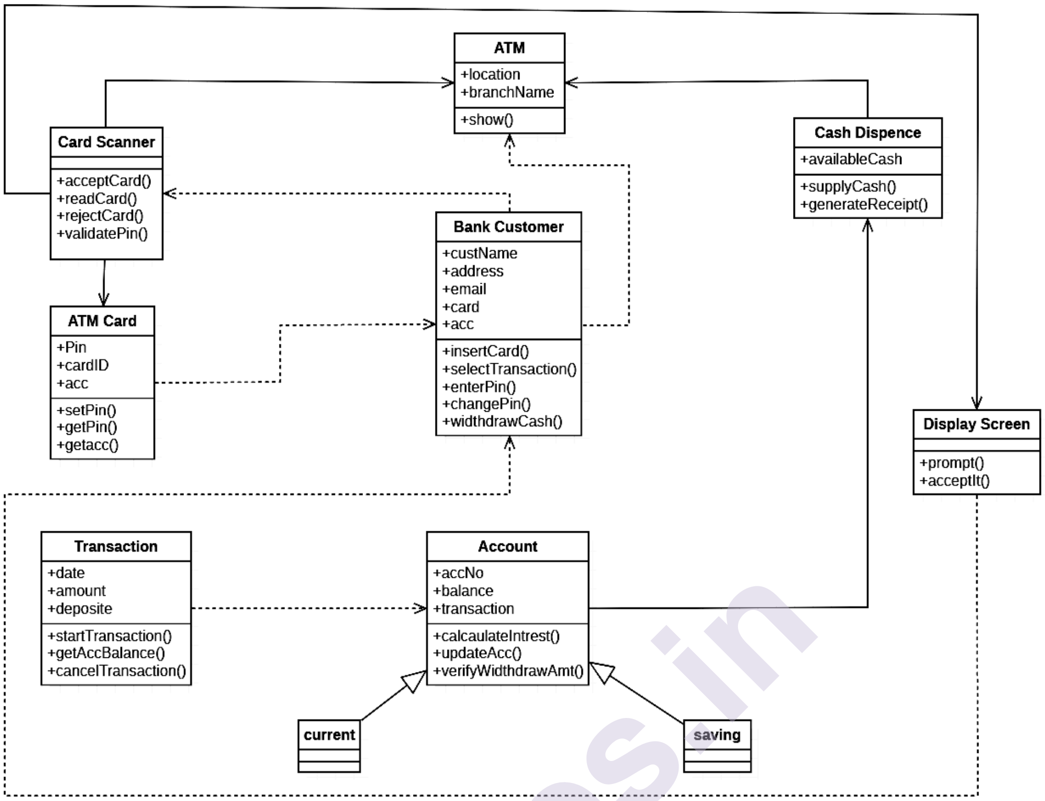


10. Realization-

- It shows the realization between two objects.
- It is shown between interfaces and the class or component.
- It is represented by a dashed line with a hollow arrowhead.
- Symbol shown as



Class diagram Example –ATM system



4.5.2 Object Diagram

- It is an instance of a class diagram.
- It shows the snapshot of detailed state of the system.
- It is used to depict the set of object and their relationship as an instance.

Purpose of Object Diagram-

- It is used to represent the data structure examples.
- Object diagram treated as test cases that will used to check accuracy and completeness of class diagram.
- It is used to identify fact about particular element and link.

How to draw Object Diagram-?

The following points should be remembered while drawing a class diagram –

1. Diagram should have meaningful name
2. Elements along with data are to be identified.
3. Association should be clearly mentioned.
4. Different element values need to be captured.
5. For more clarity add notes.

Basic Object Diagram Symbols and Notations

1. Object Names-

- Object shown by rectangle.
- Object name and class name are underline and separated by colon.

- Symbol-

Object name: Class name

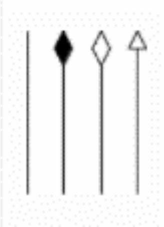
2. Object attributes-

- Here similar to class objects are having attributes.
- Object attributes should have values assign to them.
- Symbol

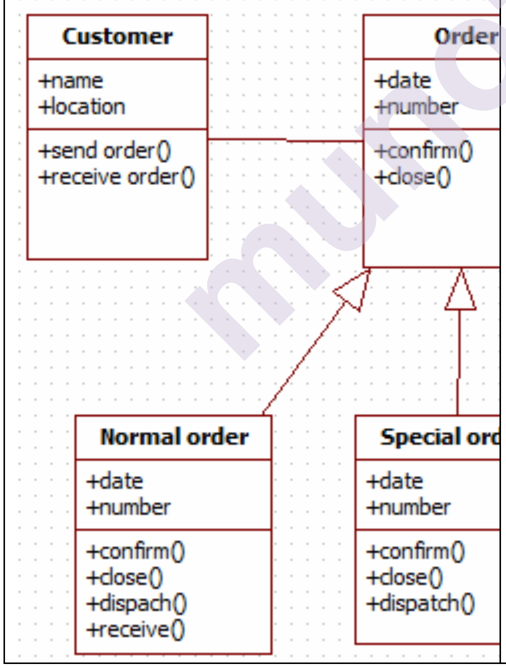
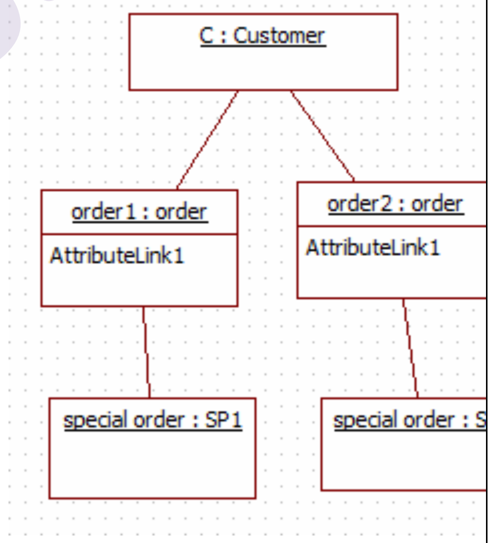
Object: class name
Attribute: value

3. Links-

- Links tend to be instances associated with associations.
- Symbols



Example of class to object diagram-

Class diagram	**Object Diagram for given class diagram**
	

4.5.3 Use Case Diagram

- It is primary form for new software program which is under developed.
- It specifies exact behavior of system.
- It gives the representation in both ways textual and visual
- Helps to design the system from user's perspective.

Purpose of Use-Case Diagram

- To capture dynamic aspect of the system.
- T0 capture, describe and document the requirements.
- To get outside view of system.
- Finds internal and external factors affecting system
- Specify the context of a system
- Drive implementation and generate test cases

How to draw Use Case Diagram?

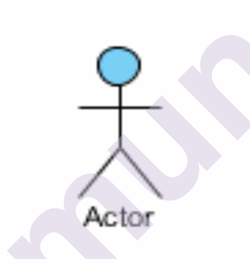
The following points should be remembered while drawing a Use Case diagram –

1. Use case name should be simple and clearly define functionality.
2. Actor name must be simple and clear.
3. Show relationship and dependency clearly.
4. Use notes to add more clarification.

Basic Use Case Diagram Symbols and Notations

1. An Actor

- ➢ Represents a role that interact with system
- ➢ An Actor can be device, person or other system
- ➢ To find an actor we have to ask some questions like
- ✓ Who uses main functionality of system?
- ✓ Which hardware device system needs to handle?
- ✓ With which the other system needs to interact?
- ✓ What nouns are used to describe system.
- ➢ The notation used for use case is a stick person or node or device symbol.



2. Use Case-

- ➢ This is an abstraction of the set of sequences. Example-login
- ➢ Name is given by verb and noun.
- ➢ Each Actor must be linked to a use case, while some use cases may not be linked to actors.
- ➢ To find Use Case We have to consider some questions like
- ✓ What function system is expected to perform?
- ✓ What do actor expect system to perform?
- ✓ What are the input and output expected by system?
- ✓ What are the verbs which used to describe the system?
- ➢ The notation used for use case is an oval with use case name written inside.

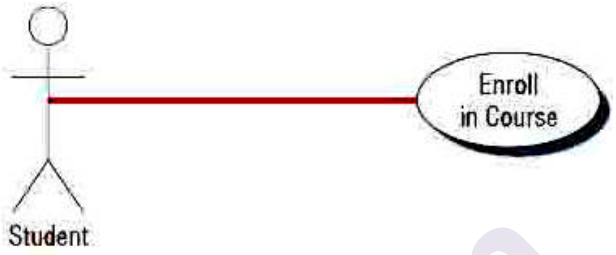
3. Relationships-

- It indicates which actor initiate which use case.
- This link indicate interaction between a specific actor and specific use case.
- The participation of an actor in a use case is shown by connecting an actor to a use case by a solid link.
- The notation used for it is

――――――――――

- **The notation used for relationship fall in to three categories**

1. Actor to Use Case



2. Between Use Case-

There are two types of relationships exist between Use Cases

a. Include-

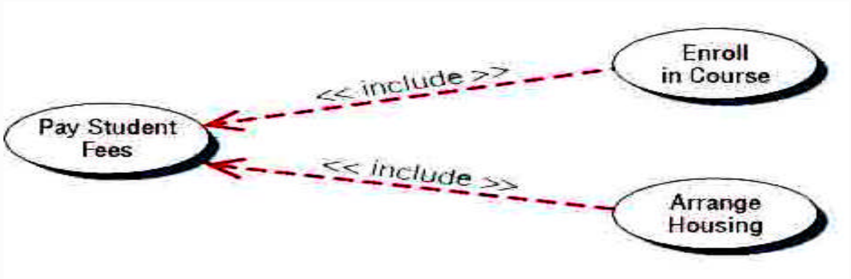
✓ When a use case is depicted as using the functionality of another use case, the relationship between the use cases is named as include or uses relationship.

✓ A use case includes the functionality described in another use case as a part of its business process flow.

✓ A uses relationship from base use case to child use case indicates that an instance of the base use case will include the behaviour as specified in the child use case.

✓ An include relationship is depicted with a directed arrow having a dotted line. The tip of arrowhead points to the child use case and the parent use case connected at the base of the arrow.

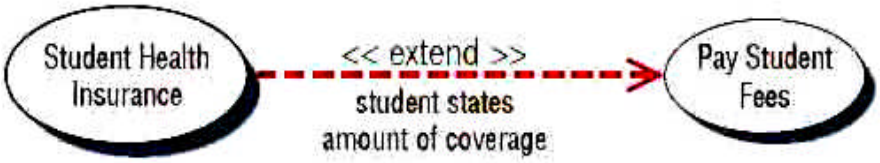
✓ The stereotype "<<include>>" identifies the relationship as an include relationship.



b. Extends-

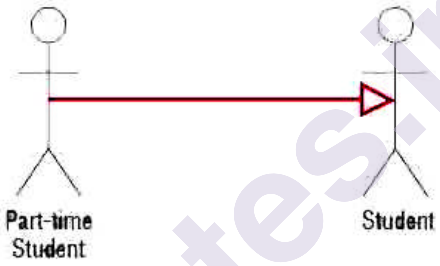
✓ It is used to relate the use cases which describe variations in the normal flow of events for the particular function to the basic use case which describe normal flow of events for the function.

- ✓ The stereotype "<<extends>>" identifies as an extend relationship



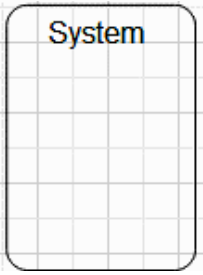
3.Between actors-

- ✓ Two actors can be related only through the generalization-specialization relationship.
- ✓ This is used if specialised actor doing some kind of interaction which is not done by rest of the actors.
- ✓ The notation used a line with arrow where the arrow head is empty triangular head.



4.Boundry of System-

- ➢ The system boundary is potentially the entire system as defined in the requirements document.
- ➢ For large and complex systems, each module may be the system boundary.
- ➢ For example, for an ERP system for an organization, each of the modules such as personnel, payroll, accounting, etc.



Example-Use Case Diagram for ATM System

??

??

??????

4.5.4 Activity Diagram

- It describes the flow of activity
- It describes dynamic aspect of the system.
- It is used to describe complex activities in high level activities.

Purpose of Activity Diagram-

- Examine business workflow and identify candidate use case
- Identify pre and post conditions
- Describe various flow of system like parallel, branched, concurrent etc.
- Modelling business requirements.

How to draw Activity Diagram?

The following points should be remembered while drawing a Activity diagram

1. Set the scope of activity that is being describe.
2. Provide proper title to diagram.
3. Identify the activities, control flow between activities
4. Identify the decision which is present in model.
5. Identify any prospect for parallelism in the process.
6. If required then add swim lanes in a diagram.

Basic Activity Diagram Symbols and Notations-

1. Initial state or start point-

- It represents start point for any activity diagram.
- In case of swim lane the start point is place in the top left corner of first column.
- This notation is small filled circle followed by an arrow.

??

2. Activity / Action state-

- Used to represent the activities of the process.
- The notation is rectangle with rounded corners.

??

3. Action Flow-

- It represents transition between activities.
- The notation is an arrowed line.

??

4. Object flow-

- It represents creation and modification of objects by activity.

??

5. Decision and branching-

- When activity requires to take a decision for moving into next activity then decision and branching is used.
- It divides the activity in true and false part.
- The notation is diamond shape.

??

6. Guards-

- These are the statements written next to decision diamond.
- These are not essentials but useful.

??

7. Synchronisation-

- In this fork and join nodes are used.
- Join node joins multiple flows in single flow.
- Fork node splits single flow in multiple flow.
- It represents by straight thicker line.

??

8. Time event-

- It stops the flows of activity for a time.

??

9. Merge Event-

- It brings multiple flows in single.

??

10. Sent and received signal

- It shows how the activity is modified from outside the system.
- It always comes in pair.

??

11. Swim lane –

- It group the activity in to columns for clear presentation.

12. Final state-

- It represents final action state.
- The notation is an arrow pointing to filled circle.

??

Example of Activity Diagram-

1. Activity Diagram for overall ATM Machine.

??

2.Activity Diagram for ATM Machine using Swim lane

??

4.5.5 Sequence Diagram

- This diagram shows interaction based on time ordering.
- It helps to understand how the different objects interact with each other.
- It represents the life line of participants while exchanging message.
- It helps to understands which classes are required for interaction.

Purpose of Sequence Diagram-

- Represents use cases in visual format.
- Used as requirement document to represents pre requisites for future implementation.
- Represents the working of object in current system.

How to draw Sequence Diagram?

1. Select only those use cases which contains complex interaction.
2. The actor should be same as use case diagram.
3. Find out the objects required for interaction.

4. Decide the flow of messages for exchanging data.
5. Write text along with message arrow.
6. Show the interaction between the objects by drawing lifelines.
7. Determine the type of interaction i.e. synchronous, asynchronous, return, creation and destruction of objects.

Basic Sequence Diagram Symbols and Notations-

1. Object-

➢ It represents the name of class.

??

2. Lifeline notation-

➢ It shows interaction of objects with each other.

➢ No two life line notations overlapped with each other.

➢ The life line Shown by vertical dashed line starting from particular actor.

??

3. Actors-

➢ It represents type of role where it interacts with system and objects.

??

4. Messages-

➢ Shows communication between objects.

➢ It represents using an arrow.

➢ Following are the categories of messages-

i. Synchronous Message-

✓ This message waits for reply before moving to next interaction i.e acknowledgment from receiver.

✓ It is represented by solid arrow head.

??

ii. Asynchronous Message-

✓ It does not wait for reply from receiver.

✓ It is showed by lined arrow head.

??

iii. Create Message-

✓ Used to instantiate a new object.

✓ It is represented with a dotted arrow and create word labelled on it <<create message>>

??

iv. Delete Message-

✓ Used to delete an object.

✓ It is represented by an arrow terminating with a x.

??

v. Self Message-

✓ Used to send message to itself.

✓ represented with a U shaped arrow.

??

vi. Replay Message-

✓ Messages are sent from sender to receiver.

✓ Represented an open arrowhead with a dotted line.

??

vii. Found Message-

✓ It is used when unknown source send message.

- ✓ Represented using an arrow directed towards a lifeline from an end point.

??

viii. Lost Message-

- ✓ Used to represent when recipient not known.
- ✓ It is represented using an arrow directed towards an end point from a lifeline

??

5. Looping-

- ➢ Some set of interaction required to perform again & again at that time looping is used.
- ➢ A box put around these steps to show looping.

6. Boundary-

- ➢ It encloses set of objects, lifelines, interactions involved in diagram
- ➢ It is represented by simple rectangle.

Example- Sequence Diagram of ATM System

??

4.5.6 State Transition Diagram

- It shows behavior of class to external environment.
- It shows dynamic behavior in between states.
- It is also called as State Diagram or State Chart Diagram.

Purpose of State Transition Diagram-

- ▪ Used to represent finite state machine.
- ▪ Used to model the objects which contain finite states and interaction with outside world.
- ▪ Describe the behavior of single object in different event.

How to draw State Transition Diagram?

1. Identify initial and final state.
2. Identify actions/events to be carried out.
3. Show the transition of object from one state to another.

Basic State Transition Diagram Symbols and Notations

1. States-

- ➢ It represents states of object during its lifetime.
- ➢ It is shown by rectangle with rounded corners.

??

2. Transition-

- ➢ It is a path between various object states.
- ➢ Transition is labeled with event that will carried out.
- ➢ Transition can come back to itself.

??

3. Initial State-

- ➢ Represents starting point of an object.

➢ It represents by filled circle followed by an arrow.

??

4. Final State-

➢ Represents final state of an object.

➢ It represents by an arrow pointing to a filled circle nested inside another circle.

??

5. Synchronization and splitting of control-

➢ It represents with small bar with two transitions.

➢ One is fork in single transition splits into multiple transitions.

➢ Second join in multiple transition combine in to single transition.

?? ??

Example-State Transition Diagram of ATM machine

1. State Transition Diagram for one Transaction ATM Machine

??

2. State Transition Diagram for one Session of ATM machine

??

4.5.7 Deployment Diagram

- It describes static view of system.
- It shows visualization of physical component of system.

Purpose of Deployment Diagram-

- They show the structure of the run-time system
- They capture the hardware that will be used to implement the system and the links between different items of hardware.
- They model physical hardware elements and the communication paths between them
- They can be used to plan the architecture of a system.
- They are also useful for Document the deployment of software components or nodes

How to draw Deployment Diagram?

1. First identify the artifact like nodes and relationship among nodes.
2. Add other elements to the diagram i.e. component, active object.
3. Add dependency between components and object when required.

Basic Deployment Diagram Symbols and Notations

1. Artifact-

✓ Represents specification of real world entity.

✓ Describe the framework during software development process.

??

2. Node-

✓ It is computational resource upon which artifacts are deployed for execution.

✓ It is represented using a node with stereotype <<device>>.

??

<< execution environment >>

- ✓ It is a node that represents an environment in which software is going to execute.
- ✓ For example, Java applications are executed in java virtual machine (JVM)).

??

Example-Deployment Diagram for ATM Machine

??

Summary

- ❖ There are variety of modeling techniques available
- ❖ The UML is one such technique which plays an important role in the object oriented software development.
- ❖ The most important elements of the object oriented method are class, object, encapsulation, polymorphism, inheritance etc.
- ❖ There are three main building blocks viz. things, relationship, diagrams.
- ❖ There are thirteen different diagram available with the UML.
- ❖ The class diagram gives static view of system.
- ❖ The main element of the class are classes and its relationships.
- ❖ The set of object and its relationship shown by Object Diagram.
- ❖ The complexity in the execution of interaction shown by Sequence Diagram.
- ❖ The flow of business functions is shown by Activity Diagram.
- ❖ For gathering requirement of the system Use Case Diagram is used.
- ❖ The transition from one state to another by triggering event shown by State Transition diagram.
- ❖ The visualization of physical components shown by Deployment Diagram.

Self-Learning Topics: Comparison of Requirements Elicitation Techniques

References-

- ❖ Unified Modeling Language User Guide by Ivar Jacobson
- ❖ UML 2 Bible by Tom Pender
- ❖ Object Oriented Modelling and Design with UML 2

Case study for Practice-

1. Draw a class diagram for the scenario given below. This scenario shows an inheritance hierarchy of a series of classes and their subclasses. It's for an imaginary application that must model different kinds of vehicles such as bicycles, motor bike and cars. All Vehicles have some common attributes (speed and colour) and common behaviour (turn Left, turn Right). Bicycle and Motor Vehicle are both kinds of Vehicle and are therefore shown to inherit from Vehicle. To put another way, Vehicle is the superclass of both Bicycle and Motor Vehicle. In our model Motor Vehicle shave engines and license plates. Attributes have been added accordingly, along with some behaviour that allows us to examine those attributes. Motor Vehicles is the base class of both Motor Bike and Car; therefore, these classes not only inherit the speed and colour properties from Vehicle, but also the additional attributes and

behaviour from Motor Vehicle. Both Motor Bike and Car have additional attributes and behaviour which are specific to those kinds of object.

2. Draw a class diagram for the scenario given below. This scenario is from system that models companies for a payroll or reporting system. Company object has properties such as name and employees list and get Name and get Employees as its behaviour. Employee object includes employee no, name, salary and manager as its properties getName (), get Emplyoee No () ,get Salary() get Manager() as its methods. get Manager() accepts object of manager. Company may have one or more employees. A manager object keeps manages as list property and add Team Member (employee_list) and get Team Member() as its behaviours. One or more employee can be managed by manager objects. Some employees are contractual employees who are within a lieu of a contractor object. A contractor object may have length_of _contract as its property and get Length() as its behaviour

3. Construct a high level sequence diagram for online bookshop. Online customer can search book catalogue, view description of a selected book, add book to shopping cart, do checkout.

4. Draw a Sequence diagram for the following scenario which shows interactions involved in using LIBSYS for downloading and printing an article. There are four objects of classes- Article, Form, Workspace and Printer--involved in this interaction. Essentially, a user request for an article triggers a request for a copyright form. Once the user has completed the form, the article is downloaded and sent to the printer. Once printing is complete, the article is deleted from the LIBSYS workspace.

5. You need to develop a web-based application in such a way that user can search other users and after getting search complete, the user can send the friend request to other users. If the request is accepted, then both users are added to friend list of each other. If one user does not accept the friend. The second user can send friend request. The user can also block each other Draw a state transition diagram for the above application

6. Draw an activity diagram from the narrative text on “ONLINE PAPER SUBMISSION SYSTEM”. The author completes an online form that requests the user to input author name, Correspondence address, email and, title of paper. The system validates this data and, if correct, asks the author to submit the paper. The author then browses to find the correct paper on their system and submits it. Once received and stored, the system returns to the author a reference number for the paper. Authors may submit as many papers as they like to be considered for acceptance to the conference up until the deadline date for submissions. Papers are allocated to referees for assessment. They review each paper and submit to the system their decision. Once the program organizer has agreed the decisions authors are informed by email. Accepted papers are then schedule to be delivered at a conference. This involves allocating a date, time and place for the presentation of the paper.

❖❖❖❖

Module - IV

5

SOFTWARE PROJECT PLANNING

Unit Structure

5.0 OBJECTIVE

- Describe and be able to prepare a business case.
- Describe the project selection process as well as the Balanced Scored approach.
- Develop a project charter and describe its relationship to the project plan.
- Describe how project planning framework links the project's measurable organizational value (MOV) to the project's scope, schedule and budget.

- Describe the difference between product scope and project scope.
- Apply several tools and techniques for defining and managing the project's scope.
- Develop a work breakdown structure.

5.1 INTRODUCTION

A business case is a deliverable that documents the project's goal, as well as several alternative or options. The feasibility, costs, benefits, and risks for each alternative are analyzed and compared, and a recommendation to approve and fund one of the alternatives is made to senior management. The first phase of the IT project methodology, as in all of phases, ends with a review of the project by the client or sponsor.

The project charter and detailed project plan make up the project's tactical plan. The project charter defines the project infrastructure and identifies the project manager, the project team, the stakeholders, and the roles each will play within the project. In addition, the project charter formalizes the project's MOV, scope, supporting processes and controls, required resources, risks, and assumptions. This project infrastructure provides the foundation for developing a detailed project plan that answers four major questions. How much will the project cost? When will the project be finished? Who will be responsible for doing the work? And, what will we ultimately achieve at the end of the project?

The term scope is used to define the work boundaries and deliverables of the project so what needs to get done, gets done-and only what needs to get done, gets done. Therefore, it is important to define not only what is part of the project is considered to be outside of the project's scope.

A work breakdown structure (WBS) is discussed first. It provides a hierarchical structure that outlines the activities or work that needs to be done in order to complete the project scope. The WBS also provides a bridge or link between the project's scope and the detailed project plan that will be entered into a project management software package.

Today, most project management software packages are relatively inexpensive and rich in features. It is almost unthinkable that anyone would plan and manage a project without such a tool. Project success, however, will not be determined by one's familiarity with a project management software package or the ability to produce nice looking reports and graphs. It is the thought process that must be followed before using the tool that counts. Thinking carefully through the activities and their estimated durations first will make the use of a project management software package much more effective. You can still create nice looking reports and graphs, but you'll have more confidence in what those reports and graphs say.

5.2 THE BUSINESS CASE

5.2.1 What is a Business Case?

A business case provides the first deliverable in the IT project life cycle. It provides an analysis of the organizational value, feasibility, costs, benefits, and risks of several proposed alternatives or options. However, a business case is not a budget or the project plan. The purpose of a business case is to provide senior management with all the information needed to make an informed decision as to whether a specific project should be funded.

For large projects, a business case may be a large, formal document. Even for smaller projects, however, the process of thinking through why a particular project is being taken on and how it might bring value to an organization is still useful.

Because assumptions and new information are sometimes used to make subjective judgments, a business case must also document the methods and rationale used for quantifying the costs and benefits. Different people who work independently to develop a business case can use the information, tools, and methods, but still come up with different recommendations.

One can also think of a business case as an investment proposal or a legal case. The business case developer has a large degree of latitude to structure arguments, select or ignore evidence, and deliver the final presentations. The outcome depends largely on the ability to use compelling facts and logic in order to influence an individual or group with decision - making authority.

A good IT business case should be

(1) Through in detailing all possible impacts, costs, and benefits;

(2) Clear and logical in comparing the costs/ benefits impact of each alternative;

(3) Objective though including all pertinent information; and

(4) Systematic in terms of summarizing the findings.

5.2.2 Developing the Business Case

The purpose of a business case is to show how an IT solution can create business value. For example, an IT project may be undertaken to:

- Reduce costs
- Create a new product or service
- Improve customer service
- Improve communication
- Improve decision making
- Create or strengthen relationships with suppliers, customers, or partners
- Improve processes
- Improve reporting capabilities
- Support new legal requirements

5.2.3 Process For Developing A Business Case

Step 1: Select the Core Team

The core team should include managers, business specialists, and users who understand the requirements to be met, as well as IT specialists who understand the opportunities, limitations, and risks associated with IT. There are several advantages for having a core team develop the business case

Credibility - A team made up of individuals from various organizational areas or departments can provide access to critical expertise and information that may not be readily accessible to others outside that particular area.

Alignment with organizational goals – Higher level managers can help connect the business case with organization's long term strategic plan and mission. This alignment may be beneficial in understanding and presenting how the expected business value of the IT project will support the overall goals and mission of the organization.

Access to the real costs – Core members with certain expertise or access to important information can help build more realistic estimates in areas such as salaries, overhead, accounting and reporting practices, training requirements, union rules and regulations, and hiring practices.

Step 2: Define Measurable Organizational Value (MOV)

The core team's objective should be to define the problem or opportunity and then identify several alternatives that will provide direct and measurable value to the organization. To provide real value to an organization, however, IT projects must align with and support the organization‘s goals, mission, and objectives.

- Be measurable – Measurement provides focus for the project team in terms of its actions. Instead of implementing an information system, the project team attempts to achieve a specific performance target.
- Provide value to organization – Resources and time should not be devoted to a project unless they provide some kind of value to the organization.
- Be agreed on – A clear and agreed on MOV sets expectations for the project stakeholders. It is important that all project stakeholders understand and agree to the project's MOV.
- Be verifiable – At the end of project , the MOV must be verified to determine if the project was a success.

The MOV guides all the decisions and processes for managing the IT project and serves as a basis for evaluating the project's achievements. In other words, a project cannot be properly planned or evaluated unless the goal of the project is clearly defined and understood.

Step 3: Identify Alternatives - Since no single solution generally exists for most organizational problems, it is imperative to identify several alternatives before dealing directly with a given business opportunity. The alternatives, or options, identified in the business case should be strategies for achieving the MOV.

The other options may provide the best solution. Consider a spectrum of choices that include:

Changing the existing business processes without investing in IT

Adopting or adapting an application developed by different area or department within the organization

Reengineering the existing system

Purchasing an off- the shelf application package from a software vendor.

Custom building a new application using internal resources or outsourcing the development to another company.

Step 4 : Define feasibility and Assess Risk –

Feasibility should focus on whether a particular alternative is worth doing. Risk – It focus on what can go wrong and what must go right.

Feasibility may be viewed in terms of:

Economic feasibility – An organization may evaluate an alternative in terms of whether funds and resources exist to support the project.

Technical feasibility – It focuses on the existing technical infrastructure needed to support the IT solution.

Organizational feasibility - It focuses on how people within the organization will adapt to this planned organizational change. How will people and way they do their jobs be impacted?

Other feasibilities – Depending on the situation and the organization, a business case may include other issues, such as legal and ethical feasibility.

Risk should focus on:

Identification – What can go wrong? What must be right ?

Assessment – What is impact of each risk?

Response - How can the organization avoid or minimize the risk ?

Step 5:Define Total cost of Ownership
It includes

- Direct or up-front costs – Initial purchase price of all hardware, software, and telecommunications equipment, all development or installation costs, outside consultant fees etc.
- Ongoing costs – Salaries, training, upgrades, supplies, maintenance etc.
- Indirect costs – initial loss of productivity, time lost by users when the system is down, the cost of auditing equipment, quality assurance, and post implementation reviews.

Step 6: Define Total Benefits of Ownership

It must include all of direct, ongoing, and indirect benefits associated with each proposed alternative.

- Increasing high -value work: - For example, a salesperson may spend less time on paper work and more time calling on customers.
- Improving accuracy and efficiency: - For example, reducing errors, duplication, or the number of steps in a process.
- Improving decision making: - For example, providing timely and accurate information.
- Improving customer service: - For example, new products or services, faster or more reliable service, convenience, and so on.

Step 7: Analyze Alternatives

Once costs and benefits have been identified, it is important that all alternatives be compared with each other consistently. There are several ways to analyze the proposed alternatives. The most common are financial models and scoring models.

Financial models – It focus on either profitability and/ or cash flows. Cash flow models focus on the net cash , may be positive or negative, and are calculated by subtracting the cash outflows from cash inflows. In general one could view the benefit associated with a particular alternative as a source of cash inflow and the costs as the source of outflows.

The most commonly used cash flow models include payback, breakeven, return on investment, net present value, and scoring.

Scoring models provide a method for comparing alternatives or projects based on a weighted score. Scoring model also allow for quantifying intangible benefits or for different alternatives using multiple criteria. Using percentage weights, one can assign values of importance to the different criteria. The weights must sum to 100 percent, and when multiplied by a score assigned to each criterion they allow a composite score that is weighted average.

Step 8 : Propose and Support the Recommendation- Once the alternatives have been identified and analyzed, the last step is to recommend one of the options. It is important to remember that a proposed recommendation must be supported. The business case should be formalized in a professional-looking report. Remember that the quality and accuracy of your work will be a reflection on you and your organization.

5.3 PROJECT SELECTION AND APPROVAL

The objective of the business case is to obtain approval and funding for a proposed alternative. However, a proposed project may have to compete against several others.

The criteria for selecting a project portfolio, a set of projects that an organization may fund, are very similar to the analysis and subsequent selection of the proposed project alternatives. An IT project portfolio mainly composed of projects with low risk or those that do not attempt to take advantage of new technology may lead to stagnation. The organization may not move ahead strategically and the IT employees may fail to grow professionally due to lack of challenge. On other hand, an organization that focuses too heavily on risky projects employing cutting-edge technology may end up in a precarious position if the IT projects experience serious problems and failures. Learning from mistakes can be useful, unless the same mistakes are repeated over and over. Thus , an organization should attempt to balance its IT project portfolio with projects that have varying degrees of risk ,cutting -edge technologies and structure.

5.3.1 The IT Project Selection Process

The selection process determines which projects will be funded in a given period . This period can be a quarter, year, or a time frame used by organization. In order to weed out projects that have little chance of being approved , many organizations use an initial screening process in which business cases submitted for review are compared with a set of organizational standards that outline minimum requirements.

Projects that meet the minimum requirements are then forwarded to a decision-making committee of senior managers who have the authority to approve and provide the resources needed to support the project.

Projects selected should then be assigned to a project manager who selects the project team and then develops a project charter and detailed plan.

5.3.2 The Project Selection Decision

Even though each project proposal should be evaluated in terms of its value to the organization, it is important to reiterate that projects should not be undertaken for technology's sake. The decision to approve a project requires a number of conditions be met:

- The project must map directly to the organization's strategies and goals.
- The project must provide measurable organizational value that can be verified at the completion of the project.
- The selection of a project should be based upon diversity of measures that include :

i. Tangible costs and benefits.

ii. Intangible costs and benefits.

iii. Various levels throughout the organization (e.g. individual , process ,department, and enterprise).

5.3.3 Balanced Scorecard approach

It helps balance traditional financial measures with operational metrics across four different prospective: finance, customer satisfaction, internal business processes, and the organization's ability to innovate and learn.

Financial perspective :-

How do we look to our shareholders or to those who provide funding to us ?

Customer perspective :-

How do we look to our customers and other stakeholders ?

Internal Processes Perspective :-

What internal processes must we excel at in order to attract and retain our customers and other key stakeholders ?

Innovation and Learning Perspective :-

How do we keep getting better ?

5.4 THE PROJECT CHARTER

The project charter and baseline project plan provide governance framework for carrying out or executing the IT project.

It serves as an agreement or contract between the sponsor and project team- documenting the project's MOV ,defining its infrastructure ,summarizing the project plan details ,defining roles and responsibilities ,showing project commitments, and explaining project control mechanisms.

(I) **Documenting the project's MOV** :- Although the project's MOV was included in the business case, it is important that MOV be clearly defined and agreed upon before developing or executing the project plan.

(II) **Defining the projects infrastructure :-**The project charter defines all of the people, resources, technology, methods, project management processes, and knowledge areas that are required to support the project. In short ,the project charter will detail everything needed to carry out the project.

(III) **Summarizing the details of the project plan** :- The project charter should summarize the scope, schedule, budget, quality objectives, deliverables ,and milestones of the project.

(IV) **Defining roles and responsibilities :-**The project charter should not only identify the project sponsor, project manager, and project team, but also when and how they will be involved throughout the project life cycle. In addition ,the project charter should specify the line of reporting and who will be responsible for specific decisions.

(V) **Showing explicit commitment to the project** :- In addition to defining the roles and responsibilities of the various stakeholders , the project charter should detail to resources to be provided by project sponsor and specify clearly who will take ownership of the project's product once project is completed.

(VI) **Setting out project control mechanisms :-** Changes to project's scope, schedule ,and budget will undoubtedly be required over the course of the project. But, the project manager can lose control and the project team can lose its focus if these changes are not managed properly.

5.4.1 WHAT SHOULD BE IN A PROJECT CHARTER

Project Identification: - It is common for all projects to have a unique name or way to identify them .It is especially necessary if an organization has several projects underway at once. Naming a project can also give the project team and stakeholders a sense of identify and ownership.

Project Stakeholders: - It is important that the project charter specifically name the project sponsor and the project manager. This reduces the likelihood of confusion when determining who will take ownership of the project's product and who will be leader of the project. In addition ,the project team should be named along with their titles or roles in the project ,their phone numbers , and e-mail addresses. This section should describe who will be involved in the project, how will be involved , and when will be involved.

Project Description: - The project charter should be a single source of information. Therefore, it may be useful to include a description of the project to help someone unfamiliar with the project understand not only the details, but the larger picture as well .This may include a brief overview or background of the project as to the problem or opportunity that became a catalyst for the project and the reason or purpose for taking on the project.

Measurable Organizational Value (MOV):- The MOV should be clear, concise, agreed on, and made explicit to the entire project stakeholders. Therefore, the project's MOV should be highlighted and easily identifiable in the project charter.

Project Scope:- The project's scope is the work to be completed . A specific section of the project charter should clarify not only what will be produced or delivered by the project team, but what will not be part of the project's scope.

The scope defined in the project charter can take the form of a narrative description of the products or services produced by the project. This narrative is often called the statement of work (SOW).

Project Schedule: - Although the details of project's schedule will be in the project plan , it is important to summarize the detail of the plan with respect to the expected start and completion dates. In addition, expected dates for major deliverables, milestones, and phases should be highlighted and summarized at a very high level.

Project Budget:- A section of the project charter should highlight the total cost of the project. The total cost of the project should be summarized directly from the project plan.

Quality Standard: - Although a quality management plan should be in place to support the project, a section that identifies any known or required quality standards should be made explicit in the project charter.

Resources: - Because the project charter acts as an agreement or contract, it may be useful to specify the resources required and who is responsible for providing those resources .Resources may include people, technology, or facilities to support the project team.

Assumptions and Risks: - Any risks or assumptions should be documented in the project charter. Assumptions may include things that may go right , such a particular team member being available for the project , or specific criteria used in developing the project plan estimates. Risks on other hand, may be thought of as anything that can go wrong or things that may impact the success of the project.

Project Charter should be summarize the following potential impacts

- **Key situations or events that could significantly impact the project's scope, schedule, or budget -** These risks, their likelihood and the strategy to overcome or minimize their impact should be detailed in the project's risk plan.

- **Any known constraints that may be imposed by the organization or project environment** -Known constraints may include such things as imposed deadlines, budgets, or required technology tools or platforms.

- **Dependencies on other projects internal or external to the organization-** In most cases, an IT project is one of several being undertaken by an organization .Dependencies between projects may exists, especially if different application systems or technology platforms must be integrated.

- **Impacts on different areas of the organization** – IT projects operate in a broader environment than the project itself. As a result, the development and implementation of an IT solution will have impact on the organization. It is important to describe how the project will impact the organization in terms of disruption, downtime, or loss of productivity.

- **Any outstanding issues**– It is important to highlight any outstanding issues that need further resolution. These may be issues identified by project sponsor, the project manager, or the project team that must be addressed and agreed upon at some point during the project. They may include such things as resources to be provided or decisions regarding the features or functionality of the system.

Project Administration: - Project Administration focuses on knowledge areas, processes, and controls that will support the project. These are actually separate sub plans or strategies that make up the project management plan.

- A communication plan that outlines how the project's status or progress will be reported to various stakeholders. This plan also includes a process for reporting and resolving significant issues or problems as they arise.
- A scope management plan that describes how changes to the project's scope will be submitted logged and reviewed.
- A quality management plans that details how quality planning, assurance, and control will be supported throughout project life cycle.
- A change management and implementation plan that will specify how the project's product will be integrated into the organizational environment.
- A human resource plan for staff acquisition and team development.

Acceptance and Approval: - Since the project charter serves as an agreement or contract between the project sponsor and project team, it may be necessary to have key stakeholders sign off on the project charter. By signing the document, the project stakeholder show formal acceptance of the project and therefore, gives the project manager and team the authority to carry out the project plan.

References In developing the project charter and plan, the project manager may use a number of references. It is important to document these references in order to add credibility to the project charter and plan , as well as to provide a basis for supporting certain processes ,practices, or estimates.

5.5 PROJECT SCOPE DEFINITION

Developing a scope statement is a useful first step for defining the scope of project and setting a boundary. A project's scope, however, should also be defined in terms of the deliverables that the team must provide. These deliverables can be divided into project –oriented deliverables and product oriented deliverables. This separation gives the team a clearer definition of the work to be accomplished and improves the likelihood of accurately assigning resources and estimating the time and cost of completing the work.

5.5.1 Project- Oriented Scope

Project – oriented deliverables, or **scope**, support the project management and IT development processes that are defined in the information technology project methodology.

Project scope includes such things as the business case, project charter, and project plan and defines the work products of various ITPM phases.

Project – oriented deliverables also include specific deliverables such as a current systems study, requirements definition, and the documented design of the information system.

Project – oriented deliverables requires time and resources and therefore, must be part of the overall project schedule and budget. Their role is to ensure that project processes are being completed so that the project's product achieves the project's MOV and objectives. Project- oriented deliverables also provide tangible evidence of the project's progress.

5.5.2 Product – Oriented Scope

Product scope therefore focuses on identifying the features and functionality of the information system to be implemented. A useful tool for refining the scope boundary and defining what the system must do is a modeling tool called a context level **data flow diagram (DFD).**

A **context level DFD**, presents a high level representation of the system that has one process and depicts all the inflows and outflows of data and information between the system and its external entities.
Another useful tool is **Use Case Diagram**, which has been used in object – oriented world as part of **Unified Modeling Language (UML).**

5.6 PROJECT SCOPE VERIFICATION

The project's scope must be verified and formally accepted by the project sponsor and other appropriate stakeholders. It is scope management process that provides a mechanism for ensuring that the project deliverables are completed.

- MOV – Is project's MOV clearly defined and agreed upon? Failure to define and agree upon the MOV could result in scope changes later in project, which can lead to added work impacting the project's schedule and budget.
- Deliverables – Are the deliverables tangible and verifiable? Do they support the project's MOV?
- Quality Standards - Are controls in place to ensure that the work was not only completed, but completed to meet specific standards?
- Milestones – Milestones tell us that a deliverable was not only completed, but reviewed and accepted.
- Review and acceptance- Are both sides clear in their expectations? The project's scope must be reviewed and accepted by the project stakeholders. The project sponsor must formally accept the boundary, product to be produced, and the project –related deliverables. The project team must be clear on what it must deliver. In both cases, expectations must be realistic and agreed upon.

5.7 SCOPE CHANGE CONTROL

It is concerned with managing actual changes to the project's scope as when they occur, to ensure that any changes to the project's scope will be beneficial.

Scope grove- It describes a project team's inability to define the project's scope. This situation is common early in a project when the project team and sponsor have trouble understanding what the project is supposed to accomplish. Scope grove can be minimized by having a clearly defined MOV and following or applying the processes, concepts, and tools.

Scope creep– It is adding features to the system once the scope of the project has been approved.It does not always come from the project sponsor side. The project team itself may come across interesting or novel ideas as the project work progresses. It must be identified and controlled throughout the project because it lengthen the project schedule and, in turn, lead to cost overruns.

Scope leap- It can occur as a result of changes in environment , the business ,and the competitive makeup of the industry .Scope leap entails changing the MOV and therefore, requires that the organization rethink the value of the current project. If this change is critical, the organization may be better off pulling the plug on the current project and starting over by conceptualizing and initiating a new project.

5.8 WORK BREAKDOWN STRUCTURE

The WBS should Be Deliverable Oriented:-

The focus of a project should be to produce something, not merely on completing a specified number of activates. Although the WBS does not provide for any explicit looping, some activities may have to be repeated until the milestone is achieved. For example, software testing may uncover a number of problems or bugs that make the software system unacceptable to the client.

The WBS Should Support the Project's MOV:-

The WBS should include only tasks or activities that allow for the delivery of the project's deliverables. Before continuing with the development of the project plan, project team should ensure that the WBS allows for the delivery of the entire project's deliverables as defined in the scope. In turn, this will ensure that the project is more likely to achieve its MOV.

-0.0 EC Bank Project
 +1.0 Conceptualize and initialize project
 +2.0 Develop charter and plan
 +3.0 Analysis
 +4.0 Design
 +5.0 Construction

-6.0 Testing
 +6.1 Test Plan

-6.2 Test results report
6.2.1 Review test plan with client
6.2.2 Carry out test plan
6.2.3 Analyze results
6.2.4 Prepare test results report and presentation
6.2.5 Present test results to client
6.2.6 Address any software issues or problems
6.2.7 Milestone: Client signs off on test results
+6.3 Milestone: Testing

+7.0 Implementation
+8.0 Close project
+9.0 Evaluate project success

Figure 1.1 - Work Breakdown Structure

The Level of Detail Should Support Planning and Control:-

The WBS provides a bridge between the project's scope and project plan- that is, the schedule and budget. Therefore, the level of detail should support not only the development of the project plan but allow the project manager and project team to monitor and compare the project's actual progress to the original plan's schedule and budget.

Developing the WBS Should Involve the People Who Will Be Doing the Work:-One way to ensure that the WBS has the appropriate level of detail to ensure that the people who do the work are involved in its development. A person who has experience and expertise in a particular area probably has a better feel for what activities need to be performed in order to produce a particular project deliverable.

Learning Cycles and Lessons Learned can Support the Development of a WBS:-

By using the concept of learning cycles, the project team can focus on what they know (the facts) ,what they think they know (assumption) , and what they need to find out (research) in order to develop a more useful WBS. Lessons learned from previous projects can be helpful in ensuring that the WBS and subsequent project plan are realistic and complete.

5.9 SOFTWARE PROJECT ESTIMATION

Estimation of the size of software is an essential part of Software Project Management. It helps the project manager to further predict the effort and time which will be needed to build the project. Various measures are used in project size estimation. Some of these are:

- Lines of Code
- Number of entities in ER diagram
- Total number of processes in detailed data flow diagram
- Function points

1. Lines of Code (LOC): As the name suggest, LOC count the total number of lines of source code in a project. The units of LOC are:

- KLOC- Thousand lines of code
- NLOC- Non comment lines of code
- KDSI- Thousands of delivered source instruction

The size is estimated by comparing it with the existing systems of same kind. The experts use it to predict the required size of various components of software and then add them to get the total size.

Advantages:

- Universally accepted and is used in many models like COCOMO.
- Estimation is closer to developer's perspective.
- Simple to use.

Disadvantages:

- Different programming languages contains different number of lines.
- No proper industry standard exist for this technique.
- It is difficult to estimate the size using this technique in early stages of project.

2. Number of entities in ER diagram: ER model provides a static view of the project. It describes the entities and its relationships. The number of entities in ER model can be used to measure the estimation of size of project. Number of entities depends on the size of the project. This is because more entities needed more classes/structures thus leading to more coding.

Advantages:

- Size estimation can be done during initial stages of planning.
- Number of entities is independent of programming technologies used.

Disadvantages:

- No fixed standards exist. Some entities contribute more project size than others.
- Just like FPA, it is less used in cost estimation model. Hence, it must be converted to LOC.

3. Total number of processes in detailed data flow diagram: Data Flow Diagram (DFD) represents the functional view of a software. The model depicts the main processes/functions involved in software and flow of data between them. Utilization of number of functions in DFD to predict software size. Already existing processes of similar type are studied and used to estimate the size of the process. Sum of the estimated size of each process gives the final estimated size.

Advantages:

- It is independent of programming language.
- Each major process can be decomposed into smaller processes. This will increase the accuracy of estimation

Disadvantages:

- Studying similar kind of processes to estimate size takes additional time and effort.
- All software projects are not required to construction of DFD.

4. Function Point Analysis: In this method, the number and type of functions supported by the software are utilized to find FPC(function point count). The steps in function point analysis are:

- Count the number of functions of each proposed type.
- Compute the Unadjusted Function Points(UFP).
- Find Total Degree of Influence(TDI).
- Compute Value Adjustment Factor(VAF).
- Find the Function Point Count(FPC).

The explanation of above points given below:

- **Count the number of functions of each proposed type:** Find the number of functions belonging to the following types:
- External Inputs: Functions related to data entering the system.
- External outputs:Functions related to data exiting the system.
- External Inquiries: They leads to data retrieval from system but don't change the system.
- Internal Files: Logical files maintained within the system. Log files are not included here.
- External interface Files: These are logical files for other applications which are used by our system.
- **Compute the Unadjusted Function Points(UFP):** Categorise each of the five function types as simple, average or complex based on their complexity. Multiply count of each function type with its weighting factor and find the weighted sum. The weighting factors for each type based on their complexity are as follows:

Function type	Simple	Average	Complex
External Inputs	3	4	6
External Output	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

- **Find Total Degree of Influence:** Use the '14 general characteristics' of a system to find the degree of influence of each of them. The sum of all 14 degrees of influences will give the TDI. The range of TDI is 0 to 70. The 14 general characteristics are: Data Communications, Distributed Data Processing, Performance, Heavily Used Configuration, Transaction Rate, On-Line Data Entry, End-user Efficiency, Online Update, Complex

Processing Reusability, Installation Ease, Operational Ease, Multiple Sites and Facilitate Change.

Each of above characteristics is evaluated on a scale of 0-5.

- **Compute Value Adjustment Factor(VAF):** Use the following formula to calculate VAF
  VAF = (TDI * 0.01) + 0.65

- **Find the Function Point Count:** Use the following formula to calculate FPC
  FPC = UFP * VAF

Advantages:

- It can be easily used in the early stages of project planning.
- It is independing on the programming language.
- It can be used to compare different projects even if they use different technologies (database, language etc).

Disadvantages:

- It is not good for real time systems and embedded systems.
- Many cost estimation models like COCOMO uses LOC and hence FPC must be converted to LOC.

5.10 COCOMO MODEL

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry Boehmin 1981. The model uses a basic regression formula, with parameters that are derived from historical project data and current project characteristics. It predicts the efforts, development time, average team size and effort required to develop a software project**.**

COCOMO applies to three classes of software projects:

- Organic projects - This suits small software team since it has a generally stable development environment. The problem is well understood and has been solved in the past. (In short, "small" teams with "good" experience working with "less than rigid" requirements.)

Example: Small data processing or Inventory management system.

- Semi-detached projects - The software project which requires more experience and better guidance and creativity. For example, Compiler or different Embedded System (In short, "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements.)

Example: Database design or OS development.

- Embedded projects - The project with operating tight constraints and requirements is under this category. The developer requires high experiences and has to be creative to develop complex models. (In Short, developed within a set of "tight" constraints (hardware, software, operational,)

Example: Banking software or Traffic light control software.

Types of COCOMO model

COCOMO model allows software manager to decide how detailed they would like to conduct the cost estimation for their own project. COCOMO can unveil the efforts and schedule of a software product based on the size of the software in different levels. There are basic COCOMO, Intermediate COCOMO, and Detailed/Completed COCOMO.

- **Basic COCOMO model**

The basic COCOMO is used for rough calculation which limits accuracy in calculating software estimation. This is because the model solely considers based on lines of source code together with constant values obtained from software project types rather than other factors which have major influences to Software development process as a whole.

- **Intermediate COCOMO model**

Intermediate COCOMO model is an extension of the Basic COCOMO model which includes a set of cost drivers into account in order to enhance more accuracy to the cost estimation model as a result.

Classification of Cost Drivers and their attributes:

(i) Product attributes –

- Required software reliability extent
- Size of the application database
- The complexity of the product

(ii) Hardware attributes –

- Run-time performance constraints
- Memory constraints
- The volatility of the virtual machine environment
- Required turnabout time

(iii) Personnel attributes –

- Analyst capability
- Software engineering capability
- Applications experience
- Virtual machine experience
- Programming language experience

(iv) Project attributes –

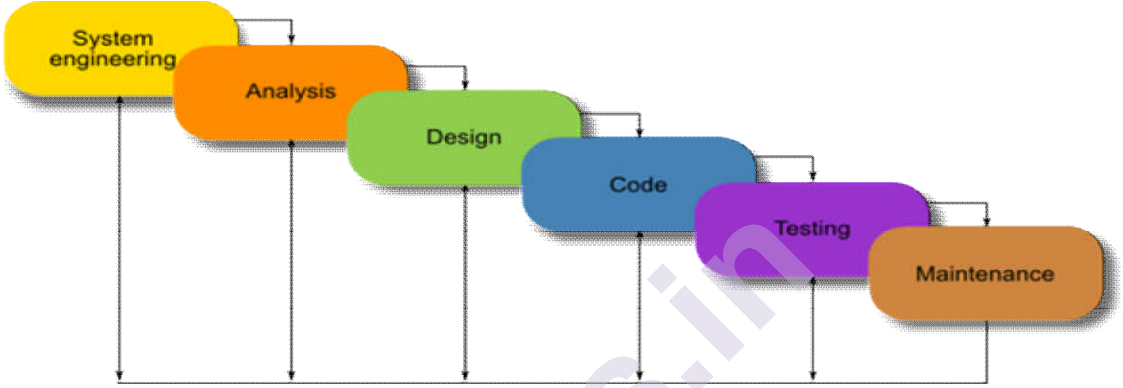
- Use of software tools
- Application of software engineering methods
- Required development schedule

- **Complete/detailed COCOMO model**

  The model incorporates all qualities of both Basic COCOMO and Intermediate COCOMO strategies on each software engineering process. The model accounts for the influence of the individual development phase (analysis, design, etc.) of the project.

The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software lifecycle.

For instance, detailed COCOMO will perform cost estimation in each development phase of Waterfall Model.



Numericals:

In COCOMO, the general calculation steps of COCOMO-based cost estimation are the following:

1. Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code KDLOC.
2. Determine a set of 15 cost factors from various attributes of the project.
3. Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in previous steps.

Example:

1. **Basic COCOMO model**

Suppose the project was estimated to be 300 KDLOC, calculate the effort, development time, and time of each of the 3 modes

Note: the basic COCOMO is used in Organic mode by default.

Here, The estimated effort and scheduled time for the project are given by the relation:

Effort (E) = a*(KLOC)^b MM

Scheduled Time (D) = c*(E)^d Months(M)

Where,

- **E** = Total effort required for the project in Man-Months (MM).
- **D** = Total time required for project development in Months (M).
- **KLOC** = the size of the code for the project in Kilo lines of code.
- **a, b, c, d** = The constant parameters for a software project.

PROJECT TYPE	**a**	**b**	**c**	**d**
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Solution:

Given estimated size of project is: 300 KLOC

- For Organic

Effort (E) = a(KLOC)^{b} = 2.4*(300)$^{1.05}$ = 957.61 MM*
Scheduled Time (D) = c(E)^{d} = 2.5*(957.61)$^{0.38}$ = 33.95 Months(M)*
Avg. Resource Size = E/D = 957.61/33.95 = 28.21 Mans
Productivity of Software = KLOC/E = 300/957.61 = 0.3132 KLOC/MM = 313 LOC/MM

- For Semidetached

Effort (E) = a(KLOC)^{b} = 3.0*(300)$^{1.12}$ = 1784.42 MM*
Scheduled Time (D) = c(E)^{d} = 2.5*(1784.42)$^{0.35}$ = 34.35 Months(M)*

- For Embedded

Effort (E) = a(KLOC)^{b} = 3.6*(300)$^{1.2}$ = 3379.46 MM*
Scheduled Time (D) = c(E)^{d} = 2.5*(3379.46)$^{0.32}$ = 33.66 Months(M)*

2. Intermediate COCOMO model

The estimated effort and scheduled time are given by the relationship:

Effort (E) = a*(KLOC)^{b} *EAF MM
Scheduled Time (D) = c*(E)^{d} Months(M)

Where,

- **E =** Total effort required for the project in Man-Months (MM).
- **D =** Total time required for project development in Months (M).
- **KLOC =** The size of the code for the project in Kilo lines of code.
- **a, b, c, d =** The constant parameters for the software project.

EAF = It is an Effort Adjustment Factor, which is calculated by multiplying the parameter values of different cost driver parameters. For ideal, the value is 1.

Suppose project was estimated with a size of 300 KLOC. Calculate the Effort, Scheduled time for development by considering developer having high application experience and very low experience in programming.

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Product Parameter					
Required Software	0.75	0.88	1	1.15	1.4
Size of Project Database	NA	0.94		1.08	1.16
Complexity of The Project	0.7	0.85		1.15	1.3
Hardware Parameter					
Performance Restriction	NA	NA	1	1.11	1.3
Memory Restriction	NA	NA		1.06	1.21
virtual Machine Environment	NA	0.87		1.15	1.3
Required Turnabout Time	NA	0.94		1.07	1.15
Personnel Parameter					
Analysis Capability	1.46	1.19	1	0.86	0.71
Application Experience	1.29	1.13		0.91	0.82
Software Engineer Capability	1.42	1.17		0.86	0.7
Virtual Machine Experience	1.21	1.1		0.9	NA
Programming Experience	1.14	1.07		0.95	NA
Project Parameter					
Software Engineering Methods	1.24	1.1	1	0.91	0.82
Use of Software Tools	1.24	1.1		0.91	0.83
Development Time	1.23	1.08		1.04	1.1

Solution:

Given the estimated size of the project is: 300 KLOC
Developer having highly application experience: 0.82 (as per above table)
Developer having very low experience in programming: 1.14(as per above table)
*EAF = 0.82*1.14 = 0.9348*
*Effort (E) = $a*(KLOC)^{b}$ *EAF = $3.0*(300)^{1.12}$ *0.9348 = 1668.07 MM*
Scheduled Time (D) = $c*(E)^{d}$ = $2.5*(1668.07)^{0.35}$ = 33.55 Months(M)

3. *The Detailed COCOMO*

In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:

1. Planning and requirements
2. System design
3. Detailed design
4. Module code and test
5. Integration and test
6. Cost Constructive model

Advantages and Disadvantages of COCOMO Model

Following are some advantages and disadvantages of the COCOMO model.

Advantages

- Easy to estimate the total cost of the project.
- Easy to implement with various factors.
- Provide ideas about historical projects.

Disadvantages

- It ignores requirements, customer skills, and hardware issues.
- It limits the accuracy of the software costs.
- It mostly depends on time factors.

COCOMO II Model

COCOMO-II is the revised version of the original Cocomo (Constructive Cost Model) and is developed at University of Southern California. COCOMO II takes into account different approaches to software development, reuse, etc. It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity.

COCOMO II incorporates a variety of sub-models that generate increasingly detailed software estimates .The sub-models in COCOMO II are:

a. **Application composition model:** Used when software is composed from existing parts.
b. **Early design model:** Used when requirements are available but design has not yet started.
c. **Reuse model:** Used to compute the effort of integrating reusable components.
d. **Post architecture model:** Used once the system architecture has been designed and more information about the system is available.

COCOMO II Estimation

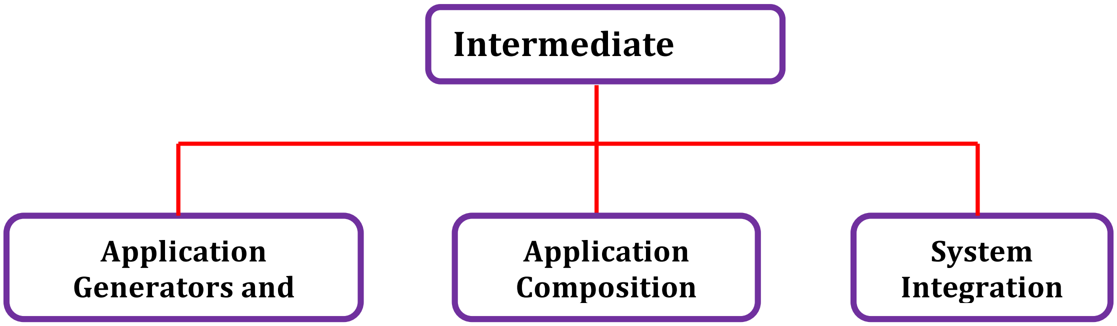
End User	Application generators & composition aids	Infrastructure
	Application Composition	
	System Integration	

1. End User Programming:

Application generators are used in this sub-model. End user writes the code by using these application generators.

Example – Spreadsheets, report generator, etc.

2. Intermediate Sector:



a. **Application Generators and Composition Aids:**

This category will create largely prepackaged capabilities for user programming. Their product will have many reusable components. Typical firms operating in this sector are Microsoft, Lotus, Oracle, IBM, Borland, Novell.

b. **Application Composition Sector:**

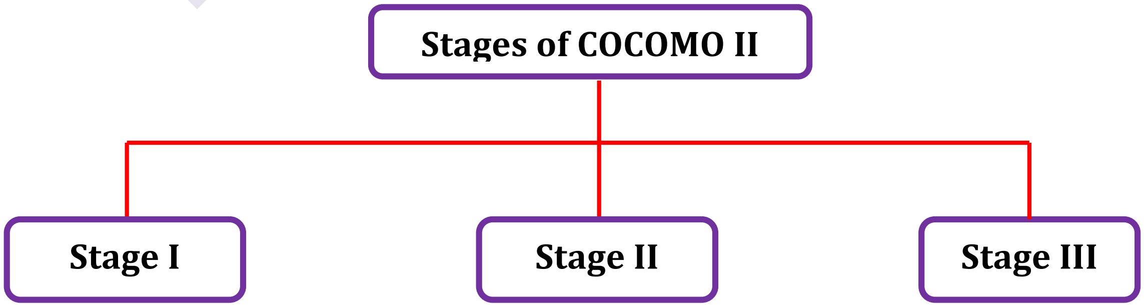
This category is too diversified and to be handled by prepackaged solutions. It includes GUI, Databases, domain specific components such as financial, medical or industrial process control packages.

c. **System Integration:**

This category deals with large scale and highly embedded systems.

3. Infrastructure Sector:

This category provides infrastructure for the software development like Operating System, Database Management System, User Interface Management System, Networking System, etc.



1. **Stage-I:**

It supports estimation of prototyping. For this it uses Application Composition Estimation Model. This model is used for the prototyping stage of application generator and system integration.

a. **Stage-II:**

It supports estimation in the early design stage of the project, when we less know about it. For this it uses Early Design Estimation Model. This model is used in early design stage of application generators, infrastructure, and system integration.

b. **Stage-III:**

It supports estimation in the post architecture stage of a project. For this it uses Post Architecture Estimation Model. This model is used after the completion of the detailed architecture of application generator, infrastructure, and system integration.

COCOMO II can be used for the following major decision situations

- Making investment or other financial decisions involving a software development effort
- Setting project budgets and schedules as a basis for planning and control
- Deciding on or negotiating tradeoffs among software cost, schedule, functionality, performance or quality factors
- Making software cost and schedule risk management decisions
- Deciding which parts of a software system to develop, reuse, lease, or purchase
- Making legacy software inventory decisions: what parts to modify, phase out, outsource, etc
- Setting mixed investment strategies to improve organization's software capability, via reuse, tools, process maturity, outsourcing, etc.

COMPARISION of COCOMO and COCZOMO-II

COCOMO I	COCOMO II
COCOMO I is convenient in the waterfall models of the software development cycle.	COCOMO II is helpful in non-sequential, rapid development and reuse models of software.
Effort equation’s exponent is determined by 3 development modes.	Effort equation’s exponent is determined by 5 scale factors.
This model is based upon the linear reuse formula.	This model is based upon the non-linear reuse formula
Development begins with the requirements assigned to the software.	It follows a spiral type of development.

5.11 EARNED VALUE MANAGEMENT

Earned value management (EVM) is a project management methodology that integrates schedule, costs, and scope to measure project performance. Based on planned and actual values, EVM predicts the future and enables project managers to adjust accordingly. It is a technique that project managers use to track the performance of their project against project baselines. Often the progress of a project is thought of simply as being ahead or behind schedule and over or under budget.

This is where knowing the earned value helps. It can provide deeper information on your project. And when learning about earned value, it's important to remember that there are three terms associated with it, which are each slightly different.

1. **Earned Value Analysis (EVA):** This project management technique is quantitative. It evaluates project performance by figuring out the likely results of the project. It does this by comparing the progress and budget of work planned to the actual costs.

2. **Earned Value Management (EVM):** This methodology measures project performance with an integrated schedule and budget, which is based on the project work breakdown structure (WBS).

3. **Earned Valued Management System (EVMS):** This is the collection of tools, templates, processes and procedures that an organization uses to do EVM.

Calculations for Earned Value Management

There are calculations that can be done quickly and easily to execute EVM. To do so, you need to know the following:

c. Planned Value (PV)

Planned value is the budgeted cost for work scheduled (BCWS). PV varies based on the scope of work in consideration and the point where you're at in the overall schedule.

PV = Total project cost * % of planned work

For example, let's say, the PV for your 5-month project is $25,000:

PV for the complete project = $25,000
PV at 2 months = $25,000 * 40% = $10,000

You can also calculate PV for a time period, say, month 2 to month 4 = $25,000 * 60% = $15,000.

d. Actual Costs (AC)

Actual costs, also referred to as actual cost of work performed (ACWP), is relatively straightforward. If you are using a robust project cost management software, tracking actual costs should not be a challenge. However, it's important to remember to include several hidden costs—material, resource, hardware, software licenses, overheads, etc.

You can look at AC cumulatively, accounting for all the activities done from the beginning of the project to date or over a specific time period. In our example, let's assume, AC at the end of 2 months = $15,000

e. Earned Value (EV)

Now, this is where EVM gets interesting. You've made a plan to finish some amount of work and budgeted accordingly. But, from experience, you know that there is bound to be some discrepancy from your estimate. At the end of 2 months, you may have planned to complete 40% of your work, but let's say you managed to just finish 30%.

The question, then, is, what's the budgeted cost for this work? EV, also called as budgeted cost for work performed (BCWP), gives you the answer.

In our example:
EV = Total project cost * % of actual work = $25,000 * 30% = $7,500

Variance Analysis

Planned value, actual cost, and earned value numbers are fundamental to variance calculations. At this point, the project manager wants to know how far off we are from the project baseline. This can be determined through schedule and cost variance.

a. Schedule Variance (SV)

Schedule variance is a quantitative indicator of your divergence from the initial planned schedule. A negative SV indicates that we are behind schedule, a positive SV indicates that we are ahead of schedule and zero means that we are exactly on schedule.
SV = EV – PV

In our example, SV at 2 months = $7,500 – $10,000 = -$2,500
SV% = (SV/PV) *100 = (-$2,500/$10,000) *100 = -25%

This implies that we are 25% behind schedule. It's interesting to note that we aim to understand schedule, a time component, from the perspective of costs. To arrive at these costs though, we needed to know the scope of work planned and completed. This is how the three pillars—scope, time and cost come together in EVM.

b. Cost Variance (CV)

Cost variance is a quantitative indicator of your divergence from the initial planned budget. A negative CV indicates that we are over budget, a positive CV indicates that we are under budget and zero means that we are exactly on budget.

CV = EV – AC
In our example, CV at 2 months = $7,500 – $15,000 = -$7500
CV% = (CV/EV) *100 = (-$7,500/$7,500) *100 = -100%
This implies that we are 100% over budget.

Again, this is an instance of how scope, time and cost come together to give you a clear picture of where you stand at the moment in your project.

5.12 SUMMARY

The business case is defines the problem or opportunity, MOV, feasibility, costs , and benefits of several alternatives that an organization may choose in order to achieve its goals and strategies. Based on the analysis of the alternatives identified , a recommendation is made to approve and fund a specific project.

The business case is formalized in a report to senior management who may review several proposed projects. The decision to fund a particular project depends on resources available and the value of the project to the organization. The balanced scorecard approach focuses on four perspectives – financial, customer, internal processes and innovation and learning . An organization should make the project selection decision based on a diverse set of measures and in terms of how well the project supports the goal and strategies of the organization.

The project charter documents the project's MOV and describes the infrastructure needed to support the project. In addition, the project charter should provide a consolidated source of information about the project and reduce confusion and misunderstanding. The project charter and project plan should be developed together – the details of project plan need to be summarized in the project charter, and the infrastructure outlined in the project charter will influence the estimates used to develop the project plan.
Product-oriented deliverables focus on the high level features and functionality of the application system – the project's product. On the other hand, project-oriented deliverables focus on the project's processes as defined in the IT project methodology.

Scope grope is a common occurrence in the early stages of the project. Often the project team struggle to define what the project is all about and what work must be done.

Scope creep, on the other hand, is a common occurrence in many projects. It entails adding additional features or functions to the scope once the scope has been set and approved. This phenomenon can increase the schedule and budget, causing the project to miss its deadline and budget targets.

Sample Questions

1. What is a business case?
2. Why should an organization develop a business case?
3. Describe the balanced scorecard approach.
4. What is the purpose of a project charter
5. How does the project charter support the project plan?
6. What is a project's scope?
7. Describe the scope definition process.

8. Describe the scope verification process.
9. Describe the scope change control process.
10. What is a WBS? What purpose does it serve?

Reference Books:

1. Software Engineering, 5th and 7thedition, by Roger S Pressman, McGraw Hill publication.
2. Managing Information Technology Project 6thedition , by Kathy Schwalbe , Cengage Learning publication.
3. Software Engineering 3rd edition by KK Agarwal , Yogesh Singh , New Age International publication.
4. Information Technology Project Management by Jack T Marchewka Wiley India publication.
5. Software Engineering for students: A Programming Approach by Douglas Bell, Pearson publication.
6. Software Engineering Project Management by Richard H. Thayer Wiley India Publication.

❖❖❖❖

Module V

6

PROJECT SCHEDULING AND PROCUREMENT MANAGEMENT

Unit Structure

6.0 OBJECTIVES OF PROJECT SCHEDULING

To arrange the manufacturing activities in such a way that the cost of production is minimized and the goods produced are delivered on due dates is the main objective of Project Scheduling. In order to meet the delivery dates the sequence of operations is properly planned. To have minimum total time of production by having better resources utilisation we use scheduling. We can also make use of this for having maximum capacity utilization and reducing the labour cost by minimization of idleness of machines and manpower. It also helps avoid unbalanced allocation of work among the various departments and workstations.

6.1 INTRODUCTION

Project Scheduling in a project is a roadmap of all activities to be done with specified order and within time slot allotted to each activity. Project managers tend to define various tasks, and project milestones and they arrange them keeping various factors in mind. They look for tasks lie in critical path in the schedule, which are necessary to complete in specific manner (because of task interdependency) and strictly within the time allocated. Arrangement of all tasks, which lies out of critical path are less likely to impact overall schedule of the project.

6.2 OVERVIEW

Scheduling in project management is the listing of activities, deliverables, and milestones within a project. A schedule also usually includes the planned start and finish date, duration, and resources assigned to each activity. Effective project scheduling is a critical component of successful time management.

For scheduling a project, following is necessary to -

Break down the project tasks into smaller, manageable form

Find out various tasks and correlate them

Estimate time frame required for each task

Divide time into work-units

Assign adequate number of work-units for each task

Calculate total time required for the project from start to finish

In fact, when people discuss the processes for building a schedule, they are usually referring to the first six processes of time management:

Plan schedule management.

Define project activities.

Sequence activities.

Estimate resources.

Estimate durations.

Develop the project schedule

6.2.1 Scheduling Principles

Compartmentalization - the product and process must be decomposed into a manageable number of activities and tasks

Interdependency - tasks that can be completed in parallel must be separated from those that must completed serially

Time allocation - every task has start and completion dates that take the task interdependencies into account

Effort validation - project manager must ensure that on any given day there are enough staff members assigned to completed the tasks within the time estimated in the project plan

Defined Responsibilities - every scheduled task needs to be assigned to a specific team member

Defined outcomes - every task in the schedule needs to have a defined outcome

(usually a work product or deliverable)

Defined milestones - a milestone is accomplished when one or more work products from an engineering task have passed quality review

6.2.2 Relationship between People and Effort

Adding people to a project after it is behind schedule often causes the schedule to slip further

The relationship between the number of people on a project and overall productivity is not linear (e.g. 3 people do not produce 3 times the work of 1 person, if the people have to work in cooperation with one another)

The main reasons for using more than 1 person on a project are to get the job done more rapidly and to improve software quality.

6.2.3 Project Effort Distribution

The 40-20-40 rule:

40% front-end analysis and design

20% coding

40% back-end testing

Generally accepted guidelines are:

02-03 % planning

10-25 % requirements analysis

20-25 % design

15-20 % coding

30-40 % testing and debugging

6.3 SCHEDULING

Task networks (activity networks) are graphic representations can be of the task interdependencies and can help define a rough schedule for particular project

Scheduling tools should be used to schedule any non-trivial project.

Program evaluation and review technique (PERT) and critical path method

(CPM) are quantitative techniques that allow software planners to identify the chain of dependent tasks in the project work breakdown structure (WBS) that determine the project duration time.

Timeline (Gantt) charts enable software planners to determine what tasks will be need to be conducted at a given point in time (based on estimates for effort, start time, and duration for each task).

The best indicator of progress is the completion and successful review of a defined software work product.

Time-boxing is the practice of deciding a priori the fixed amount of time that can be spent on each task. When the task's time limit is exceeded, development moves on to the next task (with the hope that a majority of the critical work was completed before time ran out).

6.3.1 Tracking Project Schedules

Periodic project status meetings with each team member reporting progress and problems

Evaluation of results of all work product reviews

Comparing actual milestone completion dates to scheduled dates

Comparing actual project task start-dates to scheduled start-dates

Informal meeting with practitioners to have them asses subjectively progress to date and future problems

Use earned value analysis to assess progress quantitatively

6.3.2 Tracking Increment Progress for OO Projects

Technical milestone: OO analysis complete

All hierarchy classes defined and reviewed

Class attributes and operations are defined and reviewed

Class relationships defined and reviewed

Behavioural model defined and reviewed

Reusable classed identified

Technical milestone: OO design complete

Subsystems defined and reviewed

Classes allocated to subsystems and reviewed

Task allocation has been established and reviewed

Responsibilities and collaborations have been identified

Attributes and operations have been designed and reviewed

Communication model has been created and reviewed

Technical milestone: OO programming complete

Each new design model class has been implemented

Classes extracted from the reuse library have been implemented o Prototype or increment has been built

Technical milestone: OO testing complete

The correctness and completeness of the OOA and OOD models has been reviewed

Class-responsibility-collaboration network has been developed and reviewed

Test cases are designed and class-level tests have been conducted for each class

Test cases are designed, cluster testing is completed, and classes have been integrated

System level tests are complete

6.3.3 Webapp Project Scheduling

During the first iteration a macroscopic is developed by allocating effort to specific tasks (it is understood that this is changeable schedule)

The project is broken up into increments and the increments are refined in to detailed schedules as each is begun (some increments may be developed in parallel)

Each task on the task list for each increment is adapted in one of four ways as its detailed schedule is created task is applied as is task is eliminated because it is not necessary for the increment new (custom) task is added task is refined (elaborated) into a number of named subtasks that each becomes part of the schedule

This process continues until each planned increment is completed

6.3.4 Earned Value Analysis

Earned value is a quantitative measure given to each task as a percent of project completed so far.

The total hours to complete each project task are estimated (BCWS - budgeted cost of work scheduled)

The effort to complete the project is computed by summing the effort to complete each task (BAC - budget at completion)

Each task is given an earned value based on its estimated percentage contribution to the total (BCWP - budgeted cost of work completed).

It is compute the actual cost of work performed (ACWP) at any point in the project and to compute progress indicators for both schedule and costs based on these measures

6.4 PROJECT PROCUREMENT MANAGEMENT

In project Management, Procurement is when you need to purchase, rent or contract with some external resource to meet your project goal. These relationships and any process in the project need management.

Managing these relationships means getting the best quality from the outside vendors employed by the company to assist in its doing business. There are constraints in a relationship with vendors that revolve around cost and time. Procurement management is a way to more efficiently and productively handle the process of sourcing, requisitioning, ordering, expediting, inspecting and reconciliation of procurement.

6.4.1 Process of Project Procurement Management

Project management for procurement is usually divided into four major processes:

Planning, Selection, Administering and Closing procurements.

Planning Process:

Planning, involves the creation of the official procurement management plan.

The decisions involve which items will be internally procured and which items will be externally outsourced. For every external contractor, there needs to be a statement of work (SOW) to serve as a document outlining the work being contracted. This information, will heavily impact the project's budget and financial scope.

Sample procurement documents will be prepared and criteria frameworks will be developed to create a selection of potential vendors. This selection matrix is based on the project's scope, schedule, and requirements. Risk factors and budgetary constraints are also considered.

This process is collected in the procurement management plan, which includes requirement documents, risk register, activity resource requirements, project schedule, activity cost estimates and more.

Selection Process:

The selection process involves comparing and contrasting vendors' advantages, disadvantages, and contractual offerings. Standard tools and techniques are used to select procurements, such as video conferences with bidders that allow them to understand the project requirements and ask questions.

Procurement contracts are decided and awarded through collaborations between various managers. Resource calendars are then created that detail when, where and how resources will be used and managed. The corresponding project management plan is adjusted according to resource calendar updates. Proposals are carefully evaluated and if no satisfactory bids are available, the project management team may utilize online ads to solicit new bidders.

Administration Process:

The third major step is administration, which refers to the tools and processes used to manage relationships with vendors. The administration phase results in the continual creation of procurement documents and spreadsheets that may drive project changes. A centralized system of contract change monitoring and control will be used to evaluate and determine whether potential changes to contracts are needed.

Once the contracts are signed, the management of those contractors must be folded into the overall management responsibilities. Contractors can have negative impact on budgets and schedules, which can lead to a project going off-track or worse.

It's best to contract a change control system and have regular procurement performance reviews, including inspections and audits to make sure the work is going right. Performance reporting helps keep managers informed, too. A payment system needs to be in place as well as a claims administration and a records management system.

There are formal physical inspections, internal audits and reviews of procurement operations in order to generate synthesized performance reports that provide real-time feedback. The administration process is extremely important, so it's usually managed through supply chain or project management software.

Closing Process:

Just as there is a process to start the procurement, there needs one in place to finalize it. What constitutes completed work should be detailed in the initial agreement with the contractor, so there is no confusion of either's part as to when the work is done.

The closing process isn't just about ending procurement contracts; it's about noting weaknesses, documenting successful processes and summarizing the project for future needs. Some companies prefer to conduct simple audits using performance matrices in order to grade the overall project.

Documentation is important for future projects, which may involve entirely different teams in new locations. During the closing process, negotiations may be necessary to resolve contract disputes. Ideally, potential issues will be noted during the administration process in order to begin the mediation process early.

When it comes to project procurement management, there are standard features and functions. For example, most companies prefer to use a smaller number of suppliers with long-term relationships instead of using a group of suppliers to outbid each other for the lowest price. Establishing and nurturing relationships with suppliers is important because this enables various supply chain partners and shareholders to work closely together on improvement and coordination activities.

6.4.2 Eight Steps for a Procurement Management Plan

The project manager is the project team member responsible for overseeing the procurement management plan, but it's not a one-person job. Since the procurements will be project-wide, it's important that everyone is on board with the process. Everyone should have some involvement in approving and even managing contracts.

The procurement management plan can be broken down into these eight steps.

Define Terms: To begin, start by defining the procurement terms. This means listing what you need to procure in detail: how many, what size, for how long, etc. Then you want to know what service is provided to the project and why this is important. Now add a date of use to each of these procurements and who on the project team is authorized to make these purchases.

Outline Type of Agreement: The contract is how everyone agrees on the terms of service. There are different types of contracts, for instance a fixed price and cost reimbursement are two. Therefore, the type of agreement must be decided on and how it will be managed.

Identify and Mitigate Risks: Risks are inherent in every part of a project process, and so they lie dormant in procurement until they show themselves.
It is now time to figure out what those risks might be and list them. Once a thorough list has been collected, each must have a way to resolve them. It's also good to assign a team member with the task of mitigating those risks, so they have ownership to follow through on closing them.

Define Costs: What are the costs involved with the project procurements?

Once those have been figured out, it is likely that a request for proposal will be issued, with the needs outlined and requesting bids from suppliers. Be thorough and note everything required. The suppliers will come back with their cost for products or services.

Identify Constraints: It helps to try and identify any project constraints before starting the project to avoid getting blindsided by unforeseen limitations during execution. Once this list is complete it can be looked at throughout the project phases. Constraints related to procurements include cost, scope, limited resources and technical specifications.

Get Contract Approved: Review the bids and do a service and cost analysis. Then have a list of who the decision makers are in the project group and pass the bids on to them for review. This process makes sure that everyone who needs to oversee the contract approval is involved and can provide input.

Make Decision Criteria: You have a workflow, but now you need criteria by which to decide on which bid to go into contract with. Every person who reviews the bid should have these criteria at hand to measure their response.

Create a Vendor Management Plan: Once a contract is signed, the procurement management plan will segue into a vendor management plan. The terms of the contract must be met. And, to make sure that happens, a management plan surrounding the suppliers will help ensure that goods and services are delivered as specified and on time. It is a good idea to add a performance metric to rate how well each supplier does their job, so you can improve relations on the next project and know who is worth contracting with again.

5.5 DEGREE OF RIGOR

For a project of a particular kind the degree of rigor with which the software procedure is applied may vary significantly. The degree of rigor is function of several project characteristics. For an example non-business, small, critical projects can commonly be addressed with somewhat less rigor which is large complex baseline critical applications. It should be noted however, in which all projects must be conducted in a manner which results in timely high quality deliverables.

6.5.1 Four various degrees of rigor

Four various degrees of rigor can be defined that are:

Casual: In general umbrella tasks will be documentation and minimized needs will be reduced all basic principles of software engineering are yet applicable. All procedural frame work activities are applied but only a minimum task group is needed.

Structured: Framework related tasks and activities appropriate to the project type will be applied and umbrella activities necessary to ensure high quality will be applied. The process framework will be applied for this project. SQA, SCM measurement and documentation tasks will be conducted in a streamlined manner.

Strict: - The full process will apply for this project with a degree of discipline which will ensure high quality. Robust documentation will be produced and all umbrella activities will be applied.

Quick Reaction: Framework of process will be applied for this project but because of an emergency condition only those tasks essential to maintaining good quality will be applied Back-filling example for developing a complete set of documentation conducting additional reviews will be accomplished after the application or product is delivered to the customer.

The project manager must have to develop a systematic approach for selecting the degree of rigor which is appropriate for a special project. To accomplish this project adaptation criteria are describe and a task set selector value is computed.

6.5.2 Defining a Task Set for the Software Project

Task set - collection of software engineering work tasks, milestones, and deliverables that must be accomplished to complete a particular project.
Task sets are designed to accommodate different types of projects and different degrees of rigor.

determine type of project

assess the degree of rigor required

identify adaptation criteria

Select appropriate software engineering tasks

Five Most software organizations encounter the following projects:

Concept development projects - initiated to explore some new business concept or application of some new technology.

New application development projects - undertaken as a consequence of a specific customer request.

Application enhancement projects - occur when existing software undergoes major modifications to function, performance, or interfaces that are observable by the end-user.

Application maintenance projects - correct, adapt, or extend existing software in ways that may not be immediately obvious to the end-user.

Reengineering projects - undertaken with the intent of rebuilding an existing (legacy) system in whole or in part.

5.6 CRITICAL PATH METHOD (CPM)

Critical Path Method (CPM) is a project schedule modeling technique. Mr. Morgan R. Walker and James E. Kelly developed this technique in the late 1950s.

Project planners use this method to develop schedules for many kinds of projects including IT, research, and construction.

Critical Path Method is a lengthy and complex concept. Please follow each step in this blog post and don't move on until you understand the previous steps. If you follow this advice and complete the blog post, you won't have any problems solving the questions on Critical Path Method.

A network diagram has many paths originating from one point and ending at another point. Every path has a duration and the one with the longest duration is the critical path.

You can define a critical path as:

The longest path in the network diagram, or

The shortest duration to complete the project.

6.6.1 Critical Path Method helps in:

Determining the minimum time in which the project can be completed

Determining the sequence of activities which must be completed on time in order to complete the project in time

Determining which all tasks can be delayed without delaying the project completion time

Determining the Early and Late Start of tasks

Tracking project progress with regards to agreed timeline and taking proactive corrective action if the project seems to be getting delayed

6.6.2 Technique for CPM

The technique for find out the critical path in your project can be derived as follows.

Break Down the Project: List all the tasks needed to complete the project. You can use a work breakdown structure, which is a hierarchical decomposition of the project, which includes every deliverable.

Estimate Task Duration: Now comes the tricky part, you want to know how long each task will take. If possible, get advice from others who have, so you can have the most accurate estimation of the duration of the various tasks possible.

Determine Task Dependencies: If there are any task dependencies, you want to note them, too. A task dependency is when one task cannot start until another one has been finished. It's a key element of good task management.

Add Milestones: What are the milestones in your project? Having milestones helps to keep you on track, so you can make sure you're meeting your baseline schedule.

When you have this data collected, you're able to calculate the longest path your planned tasks will take to reach the end of the project, as well as the earliest and latest that each task can start and finish without impacting the project schedule.

6.6.3 Steps for CPM

The typical steps involved to develop a project schedule using CPM method are as below:

Identify the activities for all the work packages from the project's Work

Breakdown Structure (WBS)

Sequence all the activities by identifying all the dependencies between the activities.

Develop a Schedule Network Diagram involving all the project activities ensuring that each activity has at least one predecessor and one successor except the first activity which will not have a predecessor and last activity which will not have a successor.

Estimating the duration of each activity in the schedule network diagram.

Carrying out the process of "Forward Pass" where in the "Early Start" (ES) and "Early Finish" (EF) for each activity are calculated starting from the beginning of the network diagram.

Carrying out the process of “Backward Pass” where in the “Late Finish” (LF) and “Late Start” (LS) for each activity are calculated starting from the end (Finish) of the network diagram.

Identifying the “Path” which has the longest duration in the Network Diagram. The longest path will also have the ES and LS and EF and LF of all the activities as same.

The longest path is termed as the “Critical Path”. The duration of this path will determine the shortest time taken to complete the project. Any delay on this path delays the project completion time. Hence they are critical from project’s schedule constraint point of view.

The “Non-Critical-Path” path duration will be shorter than the “Critical Path” and hence those paths will have flexibility to delay the start of the tasks on them.

The amount of time a task can be delayed on a “non-critical path” is known as “float” or “slack”, which is calculated by taking the difference between

“LS-ES” or “LF-EF”.

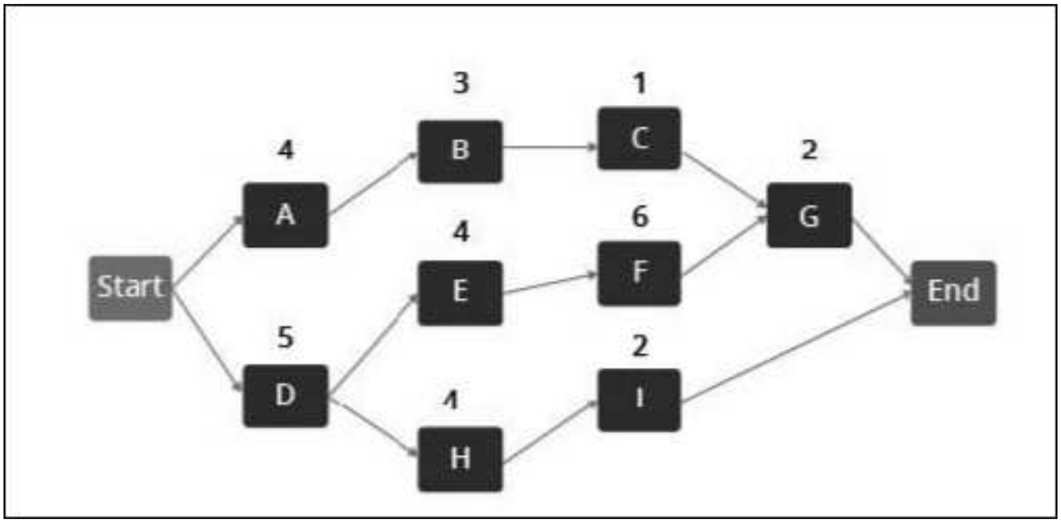
The float on critical path will be Zero to start with and not-critical paths will have a positive float time.

There may be more than one critical path in a network. But having more than one critical path increases the risk of falling behind the schedule as there are more number of tasks which if they get delayed, the project will get delayed.

6.6.4 Examples

An example of Critical Path analysis is as below (Figure 6.1).

Figure 6.1: Critical Path analysis

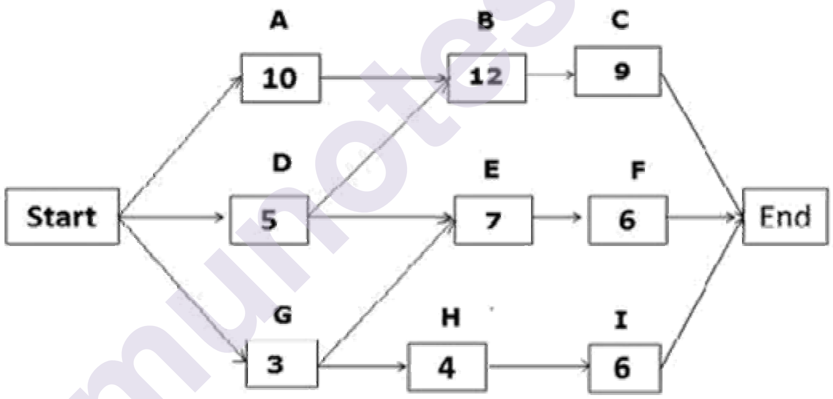


In the above diagram, the path with longest duration is Start-D-E-F-G-End is the critical path with duration of 17. The other 2 non-critical paths *Start-A-B-C-G-End* has duration of 10 and hence has a slack or float of 7 and other non-critical path *Start-D-H-I-End* has a duration of 11 and has a float of 6 days.

Critical Path once identified, the team can further explore if the duration of critical path can be compressed if the need be. Techniques such as crashing *(applying more resources on critical path)* or Fast Tracking *(doing tasks in parallel)* are applied. Compressing the critical path helps in compressing the overall project duration, thereby helping to meet the required deadline.

Based on the network diagram below (Figure 6.2), identify the total paths, critical path, and float for each path.

Figure 6.2: Total Paths



The above network diagram has five paths. The paths and their durations are as follows:

Start -> A -> B -> C-> End {duration: 31 days.}

Start ->D -> E ->F -> End {duration: 18 days.}

Start -> D -> B -> C -> End {duration: 26 days.}

Start -> G ->H ->I -> End {duration: 13 days.}

Start -> G -> E ->F -> End {duration: 16 days.}

Since the duration of the first path is the longest, it is the critical path. The float on the critical path is zero.

The float for the second path “Start ->D -> E ->F -> End” = duration of the critical path – duration of the path “Start ->D -> E ->F -> End” = 31 – 18 = 13

Hence, the float for the second path is 13 days.

Using the same process, we can calculate the float for other paths as well.

Float for the third path = 31 – 26 = 5 days.

Float for the fourth path = 31 – 13 = 18 days.

Float for the fifth path = 31 – 16 = 15 days.

Calculate Early Start, Early Finish, Late Start, and Late Finish

We have identified the critical path and the duration of the other paths. Now it’s time to move on to more advanced calculations: Early Start, Early Finish, Late Start and Late Finish.

Calculating Early Start (ES) and Early Finish (EF)

To calculate the Early Start and Early Finish dates, we use the forward pass; we will start from the beginning and proceed to the end.

The Early Start (ES) for the first activity on any path will be 1 because you cannot start an activity before the first day of your project.

The starting point for any activity is the endpoint of the predecessor activity on the same path (plus one).

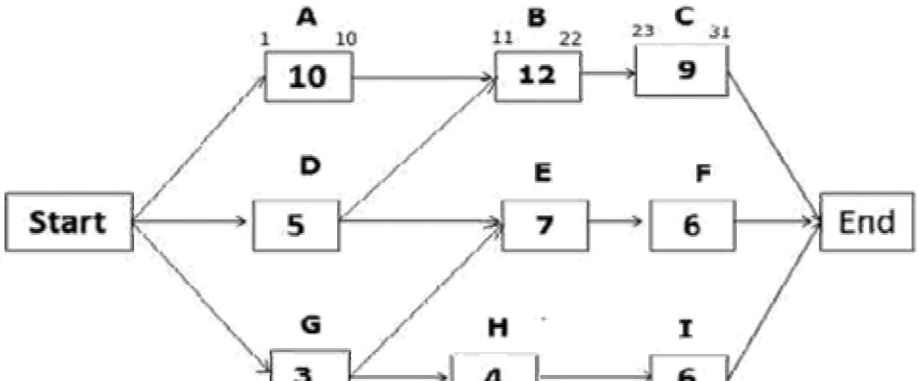
The formula used for calculating Early Start and Early Finish dates:

Early Start of the activity = Early Finish of predecessor activity + 1

Early Finish of the activity = Activity duration + Early Start of activity – 1

Early Start and Early Finish Dates for the path Start -> A -> B -> C -> End (Figure 6.3)

Figure 6.3:



Early Start of activity A = 1 (Since this is the first activity of the path)

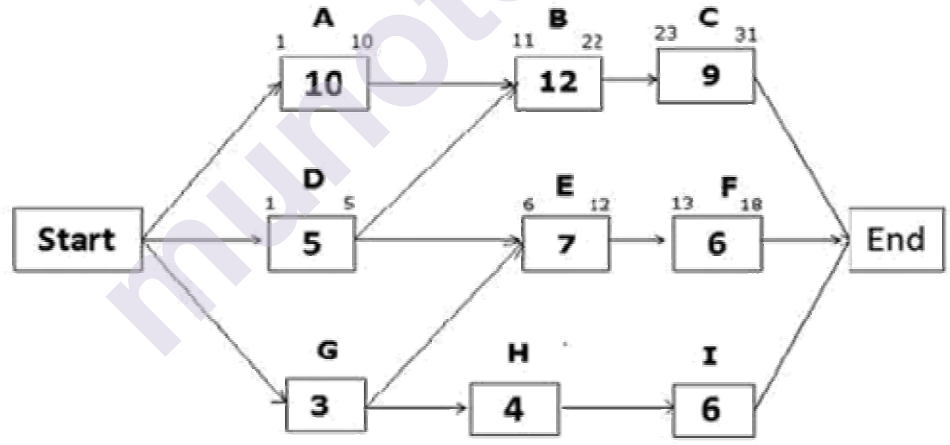
Early Finish of activity A = ES of activity A + activity duration – 1= 1 + 10 – 1 = 10 Early Start of activity B = EF of predecessor activity + 1= 10 +1 = 11

Early Finish of activity B = ES of activity B + activity duration – 1= 11 + 12 – 1 = 22 Early Start of activity C = EF of predecessor activity + 1= 22 +1 = 23

Early Finish of activity C = ES of activity C + activity duration – 1= 23 + 9 – 1 = 31

Early Start and Early Finish Dates for the path Start -> D -> E -> F -> End (Figure 64)

Figure 6.4:



Early Start of activity D = 1 (Since this is the first activity of the path)

Early Finish of activity D = 1 + 5 – 1 = 5

Early Start of activity E = EF of predecessor activity + 1

Since activity E has two predecessor activities, which one will you select? The answer is the activity with the greater Early Finish date. The Early Finish of activity D is 5, and the Early Finish of activity G is 3 (we will calculate it later).

Therefore, we will select the Early Finish of activity D to find the Early Start of activity E.

Early Start of activity E = EF of predecessor activity + 1= 5 + 1 = 6

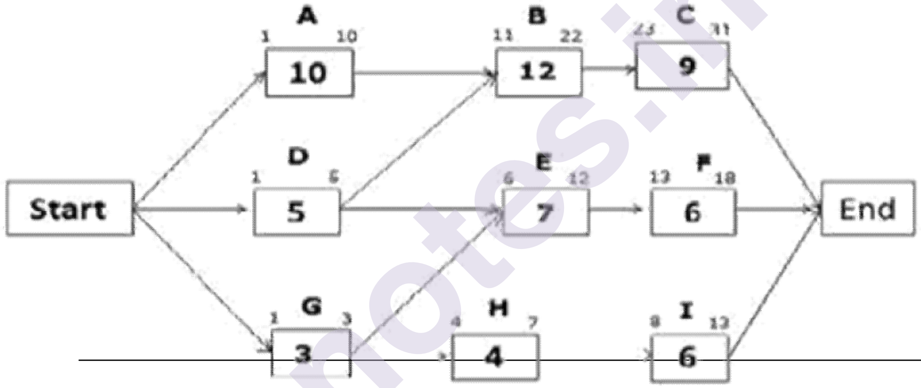
Early Finish of activity E = 6 + 7 – 1 = 12

Early Start of activity F = 12 + 1 = 13

Early Finish of activity F = 13 + 6 -1 = 18

Early Start and Early Finish Dates for the path Start -> G -> H -> I -> End (Figure 6.5)

Figure 6.5



Early Start of activity G = 1 (Since this is the first activity of the path)

Early Finish of activity G = 1 + 3 – 1 = 3

Early Start of activity H = 3 + 1 = 4

Early Finish of activity H = 4 + 4 – 1 = 7

Early Start of activity I = 7 +1 = 8

Early Finish of activity I = 8 + 6 – 1 = 13

Calculating Late Start (LS) and Late Finish (LF) We have calculated the Early Start and Early Finish dates of all activities. Now it is time to calculate the Late Start and Late Finish dates.

The Late Finish date of the last activity on all paths will be the same because no activities can continue once the project is completed.

The formula used for Late Start and Late Finish dates:

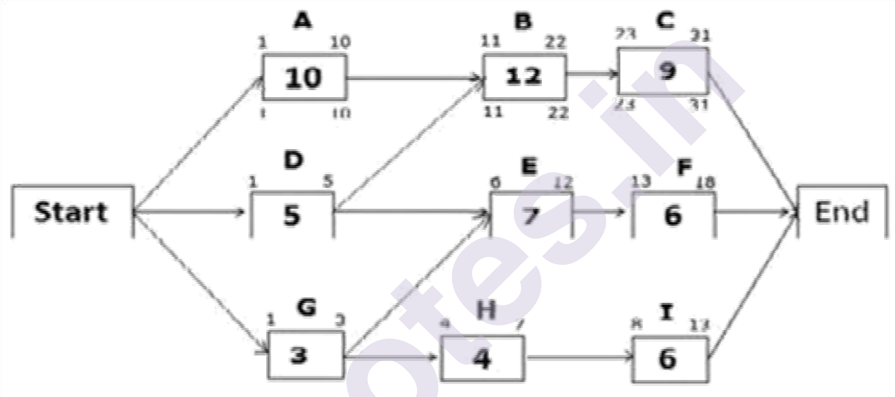
Late Start of Activity = Late Finish of activity – activity duration + 1

Late Finish of Activity = Late Start of successor activity – 1

To calculate the Late Start and Late Finish, we use the backward pass; i.e. we will start from the last activity and move back towards the first activity.

Late Start and Late Finish Dates for the path Start -> A -> B -> C -> End (Figure 6.6)

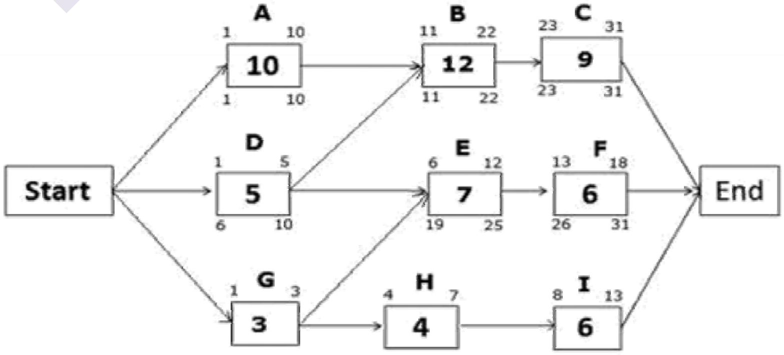
Figure 6.6:



On a critical path, the Late Start, and Late Finish dates will be the same as the Early Start and Early Finish dates

Late Start and Late Finish Dates for the path Start -> D -> E -> F -> End (Figure 6.7)

Figure 6.7:



Late Finish of activity F = 31 (because you cannot allow any activity to pass the project completion date)

Late Start of activity F = LF of activity F – activity duration + 1= 31 – 6 +1 = 26

Late Finish of Activity E = LS of successor activity – 1= LS of Activity F – 1= 26 – 1 = 25

Late Start of Activity E = LF of activity E – activity duration + 1= 25 – 7 + 1 = 19
Late Finish of activity D = LS of successor activity – 1

If you look at the network diagram, you will notice that activity D has two successor activities, B and E. So, which activity would you select?

You will select the activity with the earlier (least) Late Start date. Here, the Late Start of activity B is 11, and the Late Start of activity E is 19.

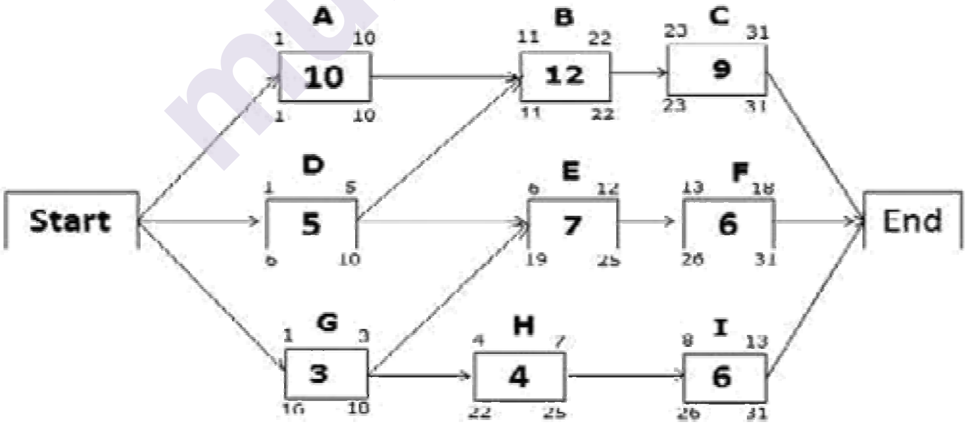
Therefore, you will select activity B, which has the earlier Late Start date. Hence,

Late Finish of activity D = LS of activity B – 1= 11 – 1 = 10

Late Start of Activity D = LF of activity D – activity duration + 1= 10 – 5 + 1 = 6

Late Start and Late Finish Dates for the path Start -> G -> H -> I -> End (Figure 6.8)

Figure 6.8:



Late Finish of activity I = 31 (because you cannot allow any activity to pass the project completion date)

Late Start of activity I = 31 – 6 + 1 = 26

Late Finish of activity H = 26 – 1 = 25

Late Start of activity H = 25 – 4 + 1 = 22

Late Finish of Activity G = 19 – 1= 18 (we will choose the late start of activity E, not activity H because the Late Start of activity E is earlier than the Late Start of activity H).

Late Start of activity G = 18 – 3 + 1= 16

6.7 SUMMARY

This Chapter includes scheduling principals and relationship between people and efforts. It describes tracking of project schedule with earned value analysis. This chapter covered the process of procurement management with some plans. After this various Degree of rigor are listed. The main part of critical path method is explained step by step with numerical.

6.8 QUESTIONS

Explain Project scheduling with the relationship between people and efforts?

State the Project Effort Distribution

Describe the Tracking Project Schedules

Explain the process of project procurement Management

What is degree of rigor?

State the Technique for CPM with example.

6.9 REFERENCE BOOKS

Information Technology Project Management by Jack T Marchewka Wiley

India publication

Software Engineering, by Roger S Pressman, McGraw Hill publication.

Software Engineering by KK Agarwal , Yogesh Singh , New Age International publication.

❖❖❖❖

Module VI

7

SOFTWARE AND SYSTEM QUALITY MANAGEMENT

Unit Structure

7.1 INTRODUCTION

Definition by ISTQB,

Quality: The degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations.

software quality: The totality of functionality and features of a software product that bear on its ability to satisfy stated or implied needs.

Software quality management (SQM) is a management process that aims to develop and manage the quality of software in such a way so as the best ensure the product meets the quality standards expected by the customer while also meeting any necessary regulatory and developer requirements, if any.

7.2 OVERVIEW OF ISO 9001

- ISO 9000-3, the Guidelines offered by the International Organization for Standardization (ISO), represent implementation of the general methodology of quality management ISO 9000 Standards to the special case of software development and maintenance.

- Both ISO 9001 and ISO 9000-3 are reviewed and updated once every 5–8 years, with each treated separately.
- As ISO 9000-3 adaptations are based on those introduced to ISO 9001, publication of the revised Guidelines follows publication of the revised Standard by a few years.
- At the time of writing, the 2000 edition of ISO 9001 (ISO, 2000a) has been issued, but only the final just-completed draft of ISO 9000-3 (ISO/IEC, 2001) is awaiting approval.
- From the 1997 edition on, the ISO 9000-3 will represent the stand-alone ISO standard for the software industry.
- The 2000 edition of ISO 9001 as well as the new edition of ISO 9000-3 are supported by two additional conceptual standards:

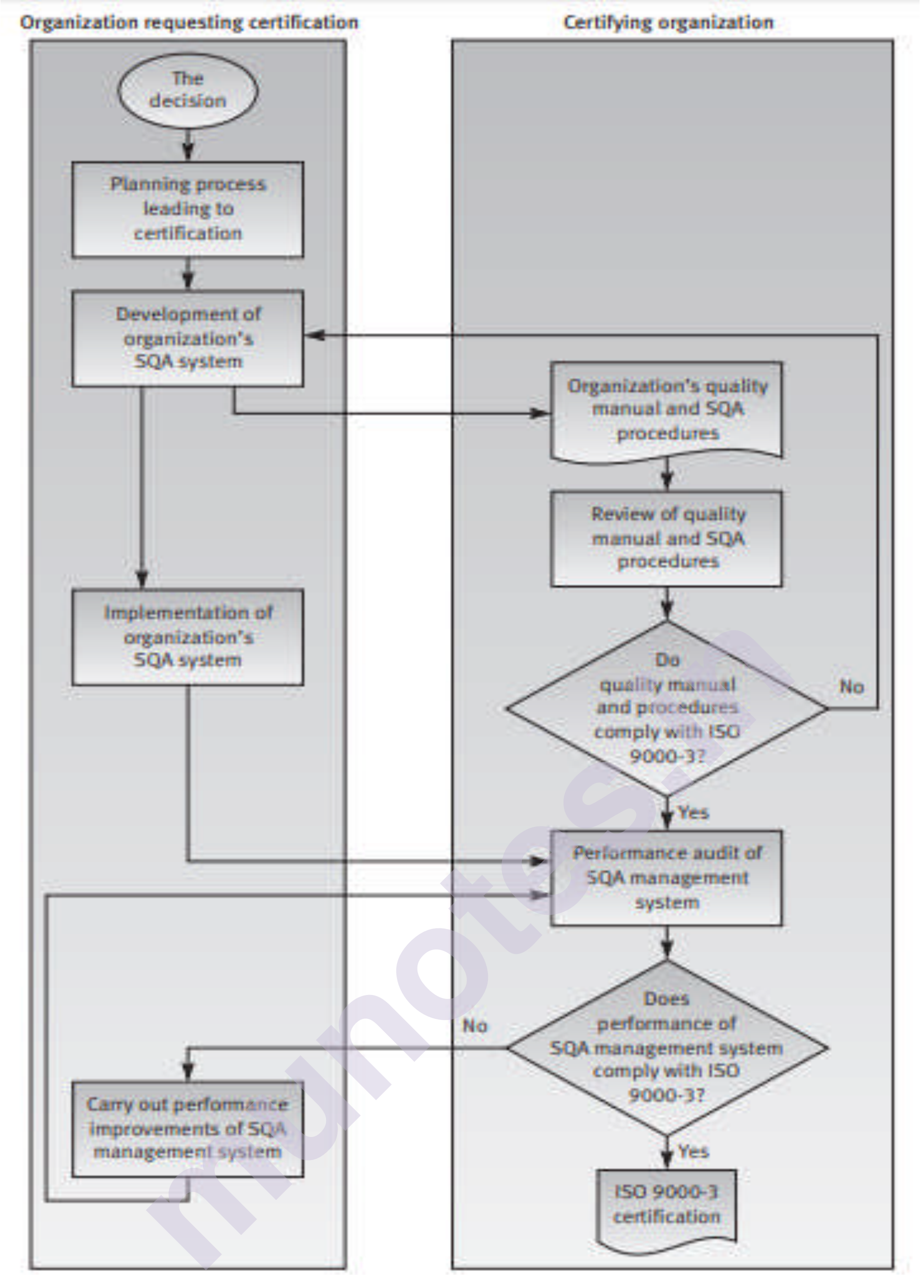
1. ISO 9000 (ISO, 2000b), which deals with fundamental concepts and terminology, and
2. ISO 9004 (ISO, 2000c), which provides guidelines for performance improvement.

ISO 9001 — application to software: the TickIT initiative :

TickIT was launched in the late 1980s by the UK software industry in cooperation with the UK Department for Trade and Industry to promote development of a methodology for adapting ISO 9001 to the characteristics of the software industry known as the TickIT initiative. TickIT is currently authorized to accredit other organizations as certification bodies for the software industry in the UK.

TickIT activities include:

- Publication of the TickIT Guide, that supports the software industry's efforts to spread ISO 9001 certification. The current guide (edition 5.0, TickIT, 2001), which includes references to ISO/IEC 12207 and ISO/IEC 15504, is distributed to all TickIT customers.
- Performance of audit-based assessments of software quality systems and consultation to organizations on improvement of software development and maintenance processes in addition to their management.
- Conduct of ISO 9000 certification audits.



Organization requesting certification
Certifying organization
The decision
Planning process leading to certification
Development of organization's SQA system
Organization's quality manual and SQA procedures
Review of quality manual and SQA procedures
Do quality manual and procedures comply with ISO 9000-3?
No
Yes
Implementation of organization's SQA system
Performance audit of SQA management system
Does performance of SQA management system comply with ISO 9000-3?
No
Yes
Carry out performance improvements of SQA management system
ISO 9000-3 certification

Figure 23.1: The ISO 9000-3 certification process

7.3 SEI CAPABILITY MATURITY MODEL

Carnegie Mellon University's Software Engineering Institute (SEI) took the initial steps toward development of what is termed a capability maturity model (CMM) in 1986.

The initial version of the CMM was released in 1992, mainly for receipt of feedback from the software community.

The first version for public use was released in 1993 (Paulk et al., 1993, 1995; Felschow, 1999).

The principles of CMM

Following are the concepts and principles of CMM assessment:

- Application of more elaborate management methods based on quantitative approaches increases the organization's capability to control the quality and improve the productivity of the software development process.
- The vehicle for enhancement of software development is composed of the five-level capability maturity model. The model enables an organization to evaluate its achievements and determine the efforts needed to reach the next capability level by locating the process areas requiring improvement.
- Process areas are generic; they define the "what", not the "how". This approach enables the model to be applied to a wide range of implementation organizations because:

➔ It allows use of any life cycle model

➔ It allows use of any design methodology, software development tool and programming language

➔ It does not specify any particular documentation standard.

The CMM and its key process areas (KPAs) are presented in following Figure

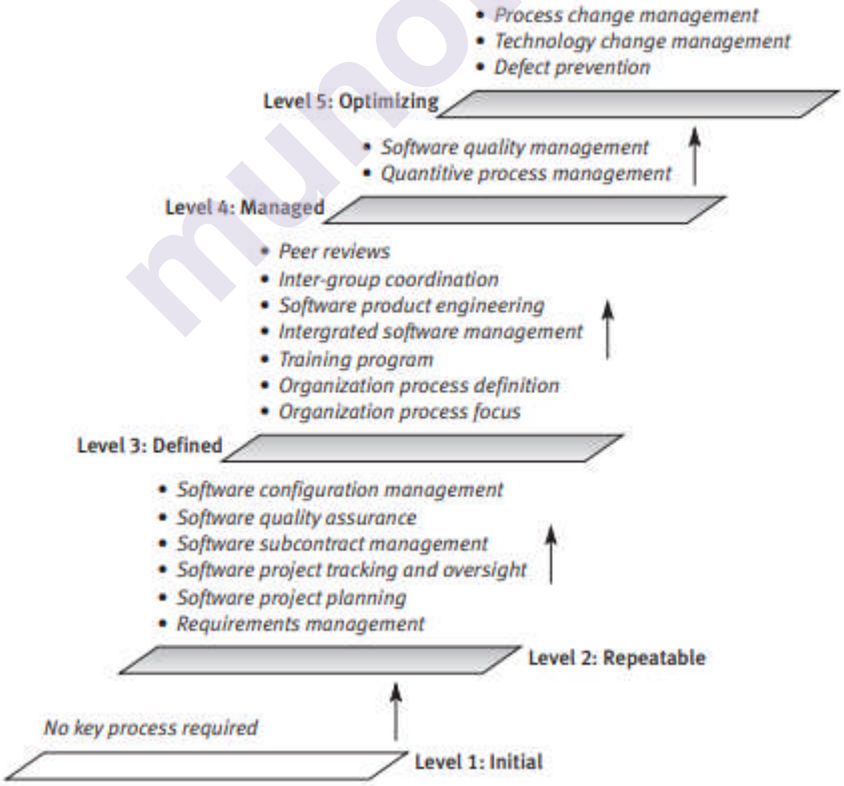

• Process change management
• Technology change management
• Defect prevention
Level 5: Optimizing
• Software quality management
• Quantitive process management
Level 4: Managed
• Peer reviews
• Inter-group coordination
• Software product engineering
• Intergrated software management
• Training program
• Organization process definition
• Organization process focus
Level 3: Defined
• Software configuration management
• Software quality assurance
• Software subcontract management
• Software project tracking and oversight
• Software project planning
• Requirements management
Level 2: Repeatable
No key process required
Level 1: Initial

Figure 23.2: The CMM model levels and key process areas (KPAs)

Source: After Paulk *et al.* (1995)

Variety of specialized capability maturity models are as following:

- **System Engineering CMM (SE-CMM)**

➔ It focuses on system engineering practices related to product-oriented customer requirements.

➔ It deals with product development: analysis of requirements, design of product systems, management and coordination of the product systems and their integration.

➔ In addition, it deals with the production of the developed product: planning production lines and their operation.

- **Trusted CMM (T-CMM)**

It was developed to serve sensitive and classified software systems that require enhanced software quality assurance.

- **System Security Engineering CMM (SSE-CMM)**

It focuses on security aspects of software engineering and deals with secured product development processes, including security of development team members.

- **People CMM (P-CMM)**

It deals with human resource development in software organizations: improvement of professional capacities, motivation, organizational structure, etc.

- **Software Acquisition CMM (SA-CMM)**

It focuses on special aspects of software acquisition by treating issues – contract tracking, acquisition risk management, quantitative acquisition management, contract performance management, etc. – that touch on software purchased from external organizations.

- **Integrated Product Development CMM (IPD-CMM)**

It serves as a framework for integration of development efforts related to every aspect of the product throughout the product life cycle as invested by each department.

7.4 MCCALLS QUALITY MODEL

The classic model of software quality factors, suggested by McCall, consists of 11 factors (McCall et al., 1977). The McCall factor model provides a practical, up-to-date method for classifying software requirements (Pressman, 2000).

McCall's Factor Model

The 11 factors are grouped into three categories – product operation, product revision, and product transition factors.

- **Product operation factors −**

  Product operation category includes five software quality factors, which deal with the requirements that directly affect the daily operation of the software. They are as follows − **Correctness, Reliability, Efficiency, Integrity, Usability.**

- **Product revision factors −**

  According to McCall's model, three software quality factors are included in the product revision category. These factors are as follows − **Maintainability, Flexibility, Testability.**

- **Product transition factors −**

  According to McCall's model, three software quality factors are included in the product transition category that deals with the adaptation of software to other environments and its interaction with other software systems. These factors are as follows − **Portability, Reusability, Interoperability.**

1 **Correctness:**

These requirements deal with the correctness of the output of the software system. They include −

- Output mission
- The required accuracy of output that can be negatively affected by inaccurate data or inaccurate calculations.
- The completeness of the output information, which can be affected by incomplete data.
- The up-to-dateness of the information defined as the time between the event and the response by the software system.
- The availability of the information.
- The standards for coding and documenting the software system.

2 **Reliability:**

Reliability requirements deal with service failure. They determine the maximum allowed failure rate of the software system, and can refer to the entire system or to one or more of its separate functions.

3 **Efficiency:**

It deals with the hardware resources needed to perform the different functions of the software system. It includes processing capabilities (given in MHz), its storage capacity (given in MB or GB) and the data communication capability (given in MBPS or GBPS).

It also deals with the time between recharging of the system's portable units, such as, information system units located in portable computers, or meteorological units placed outdoors.

4 **Integrity:**
This factor deals with the software system security, that is, to prevent access to unauthorized persons, also to distinguish between the group of people to be given read as well as write permit.

5 **Usability:**
Usability requirements deal with the staff resources needed to train a new employee and to operate the software system.

6 **Maintainability:**
This factor considers the efforts that will be needed by users and maintenance personnel to identify the reasons for software failures, to correct the failures, and to verify the success of the corrections.

7 **Flexibility:**

This factor deals with the capabilities and efforts required to support adaptive maintenance activities of the software.

These include adapting the current software to additional circumstances and customers without changing the software.

This factor's requirements also support perfective maintenance activities, such as changes and additions to the software in order to improve its service and to adapt it to changes in the firm's technical or commercial environment.

8 **Testability:**

Testability requirements deal with the testing of the software system as well as with its operation.

It includes predefined intermediate results, log files, and also the automatic diagnostics performed by the software system prior to starting the system, to find out whether all components of the system are in working order and to obtain a report about the detected faults.

Another type of these requirements deals with automatic diagnostic checks applied by the maintenance technicians to detect the causes of software failures.

9 **Portability:**
Portability requirements tend to the adaptation of a software system to other environments consisting of different hardware, different operating systems, and so forth.
The software should be possible to continue using the same basic software in diverse situations.

10 **Reusability:**

This factor deals with the use of software modules originally designed for one project in a new software project currently being developed.

They may also enable future projects to make use of a given module or a group of modules of the currently developed software.

The reuse of software is expected to save development resources, shorten the development period, and provide higher quality modules.

11 **Interoperability:**

Interoperability requirements focus on creating interfaces with other software systems or with other equipment firmware.

For example, the firmware of the production machinery and testing equipment interfaces with the production control software.

7.5 SIX SIGMA

- The process of producing high and improved quality output is known as Six Sigma.
- This can be done in two phases – identification and elimination.
- The cause of defects is identified and appropriate elimination is done which reduces variation in whole processes.
- Six Sigma's aim is to eliminate waste and inefficiency and increase customer satisfaction by delivering what the customer is expecting.
- It follows a structured methodology, and has defined roles for the participants.
- It is a data driven methodology, and requires accurate data collection for the processes being analyzed.
- It is about putting results on Financial Statements.

Following are few characteristics of Six Sigma:

The Characteristics of Six Sigma are as follows:

➔ **Statistical Quality Control:**

Six Sigma denotes Standard Deviation in statistics. Standard Deviation is used for measuring the quality of output.

➔ **Methodical Approach:**

The Six Sigma is a systematic approach of application in DMAIC(Design-Measure- Analyze-Improve-Control) and DMADV (Design- Measure- Analyze-Design-Verify) which can be used to improve the quality of production.

➔ **Fact and Data-Based Approach:**

The statistical and methodical method shows the scientific basis of the technique.

➔ **Project and Objective-Based Focus:**

The Six Sigma process is implemented to focus on the requirements and conditions.

➔ **Customer Focus:**

The customer focus is fundamental to the Six Sigma approach. The quality improvement and control standards are based on specific customer requirements.

➔ **Teamwork Approach to Quality Management:**

The Six Sigma process requires organizations to get organized for improving quality.

7.6 FORMAL TECHNICAL REVIEWS

"A formal review produces a ***report*** *that identifies the material, the reviewers, and the review team's judgment as to whether the product is acceptable. The* ***principal deliverable is a summary of the defects*** *found and the issues raised. The members of a formal review* ***team share responsibility for the quality of the review****, although authors are ultimately responsible for the quality of the products they create (Freedman and Weinberg 1990)."*

- The purpose of an Formal Technical Review (FTR) is to identify errors in function,logic and implementation of software.
- It is used to verify that the software under review meets its requirements.
- It ensures that the software has been represented according to predefined standards.
- It also helps to achieve software that is developed in a uniform manner and to make projects more manageable.

The FTR is actually a class of reviews that includes walkthroughs and inspections.

- A **review** is '*a process or meeting during which a software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval'*

- An **inspection** is '*a visual examination of a software product to detect and identify software anomalies, including errors and deviations from standards and specifications*'

- A **walkthrough** is '*a static analysis technique in which a designer or programmer leads members of the development team and other* *interested parties through a software product, and the participants ask questions and make comments about possible errors, violation of development standards, and other problems'*

Characteristics of FTR are:

- Well-defined process: Phases (orientation, etc.) Procedures (checklists, etc.)
- Well-defined roles: Moderator, Reviewer, Scribe, Author, etc.
- Well-defined objectives: Defect removal, requirements elicitation, etc.
- Well-defined measurements: Forms, consistent data collection, etc.

7.7 TOOLS AND TECHNIQUES FOR QUALITY CONTROL

- The control quality process is defined as the "process of monitoring and recording the results of executing the quality activities to assess performance and recommend necessary changes."
- In other words, quality control focuses on project results ensuring that they comply with the quality standards defined for the project and eliminating any causes of unsatisfactory performance.
- This process measures the details of the product results, such as deliverables or defects, and also of the project management results, such as schedule.
- Many of the techniques under the control quality process assume a working knowledge of statistical quality control, in particular the concepts of sampling and probability.
- The distinctions between attribute and variable sampling, precision and accuracy, and tolerance and control limits are fundamental components of a working knowledge of statistical quality control:

Prevention aids in identifying and avoiding potential problems so that they never enter or impact the process.

Inspection helps to identify and eliminate or correct errors so that they are not delivered to the customer.

Tolerance is a range of acceptable performance or results.

There are seven basic quality tools identified as appropriate for use in both the quality management plan and control quality processes. They are known as Ishikawa's seven basic tools of quality:

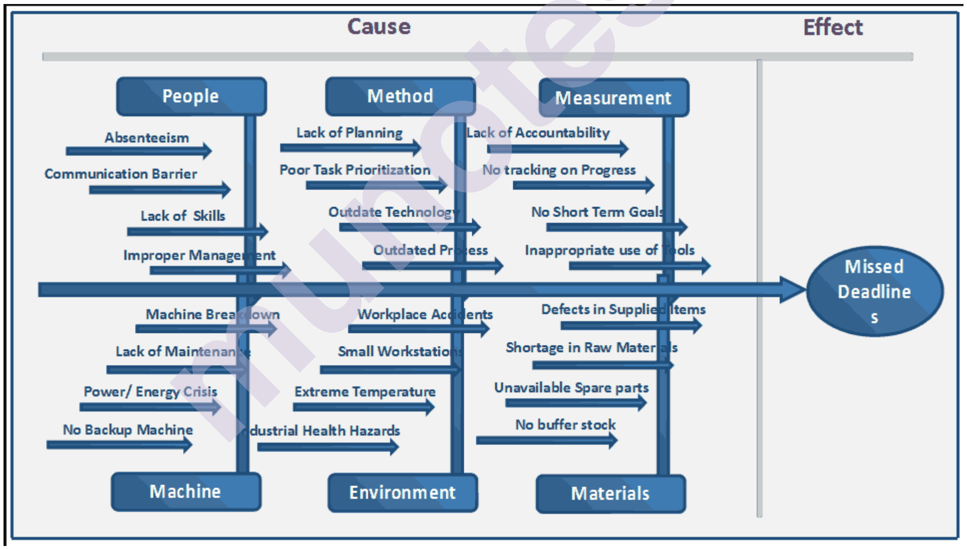
1. cause-and-effect diagrams,
2. flowcharting,
3. check sheets,
4. Pareto diagrams,
5. control charts,

6. histograms and
7. scatter diagrams.

Ishikawa's seven basic tools are also referred to as the 7QC tools.

Cause and Effect Diagrams

- ➔ Cause-and-effect diagrams, or Ishikawa diagrams, were developed by Kaoru Ishikawa to illustrate and help determine how various factors relate to potential problems.
- ➔ Also called fishbone diagrams because they resemble the skeleton of a fish.
- ➔ The head of the fish is the effect and each bone of the fish is a cause that leads to that effect.
- ➔ The bones can branch off into smaller bones as you determine the lowerlevel cause-effect relationships.
- ➔ When all the bones are filled in, the diagram lets you look at all the possible causes (individual or combinations) of the effect (or problem) so that you can develop a solution to mitigate that effect.
- ➔ The diagram allows organized thought and encourages orderly consideration of the factors that result in a certain outcome.



Flowcharts

- ➔ Flowcharts show the logical steps in a process and how various elements within a system are related.
- ➔ They can be used to determine and analyze potential problems in quality planning and quality control.
- ➔ Flowchart outlines the logical steps to complete a process.
- ➔ By documenting these logical steps, the team can identify where quality problems might occur and then develop approaches to proactively manage them.
- ➔ Flowcharting also helps create a process that is repeatable.

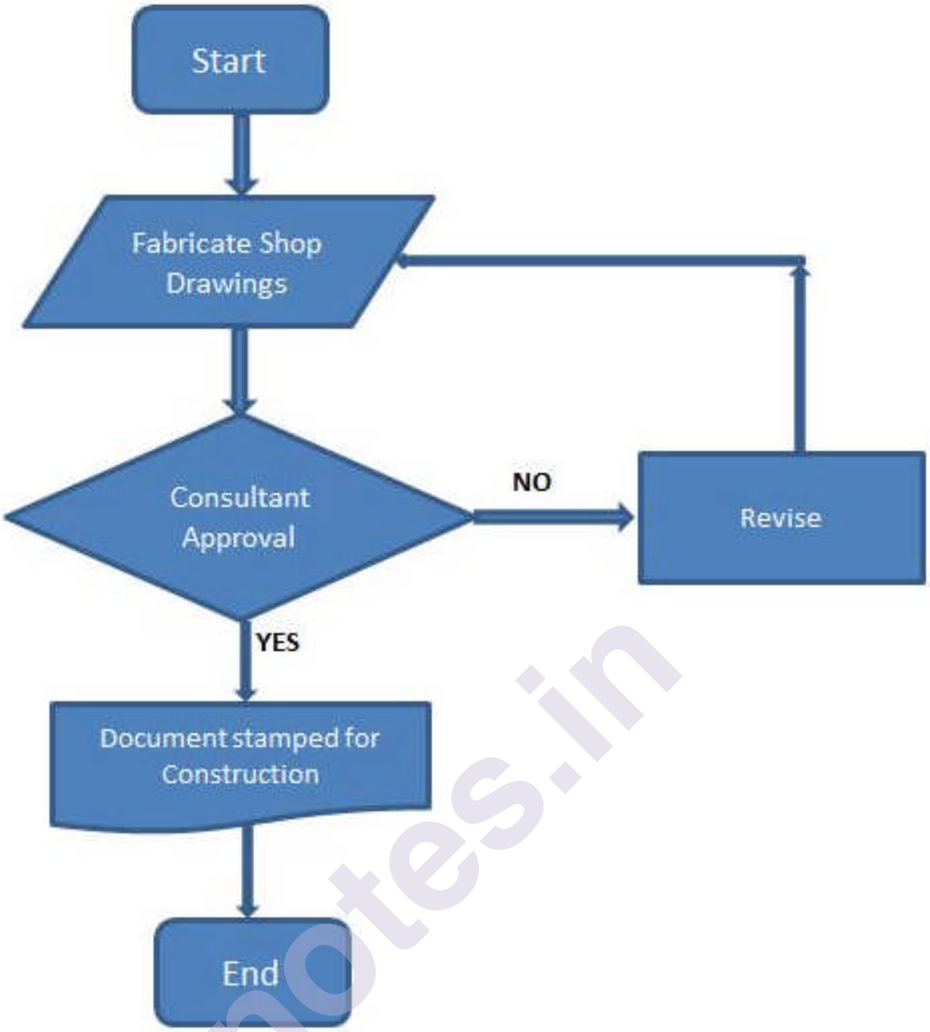
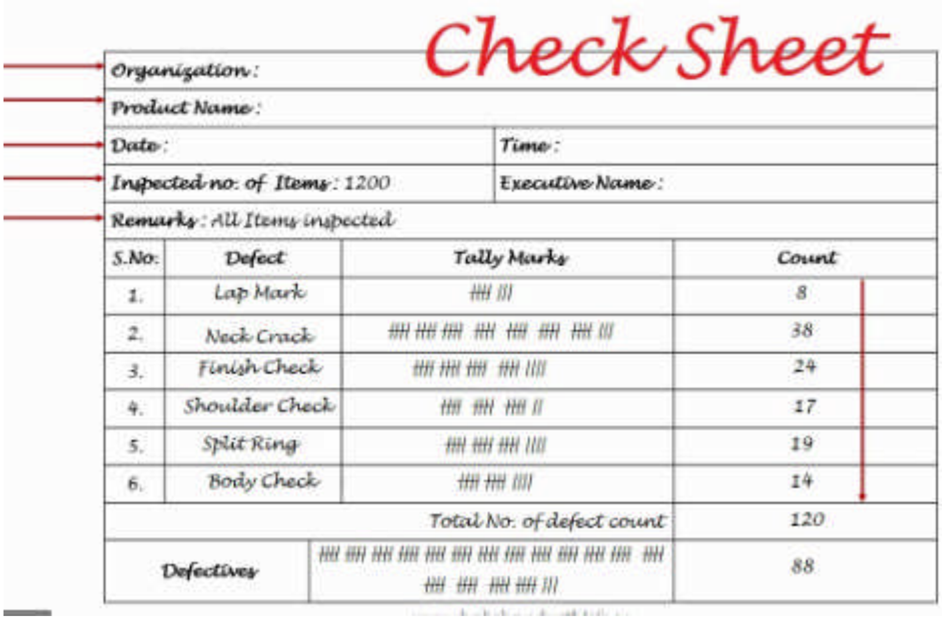


Figure A. Flowchat for drawing approval

Check Sheets

- ➔ Check sheets are used to organize information in order to facilitate data gathering.
- ➔ Check sheets are particularly effective for doing inspections, enabling focus on the particular attributes that may be contributing to potential or identified quality problems.



Check Sheet

Organization :

Product Name :

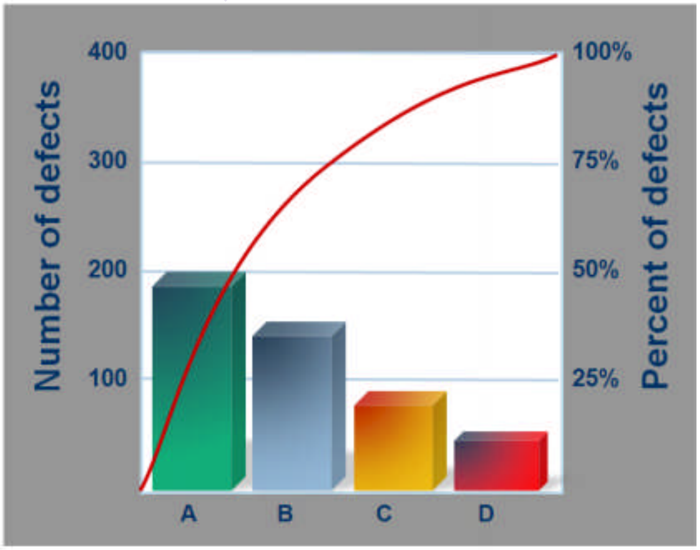
Date :	Time :
Inspected no. of Items : 1200	Executive Name :

Remarks : All Items inspected

S.No.	Defect	Tally Marks	Count
1.	Lap Mark	𝍸 |||	8
2.	Neck Crack	𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 |||	38
3.	Finish Check	𝍸 𝍸 𝍸 𝍸 ||||	24
4.	Shoulder Check	𝍸 𝍸 𝍸 ||	17
5.	Split Ring	𝍸 𝍸 𝍸 ||||	19
6.	Body Check	𝍸 𝍸 ||||	14
		Total No. of defect count	120
Defectives		𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 𝍸 |||	88

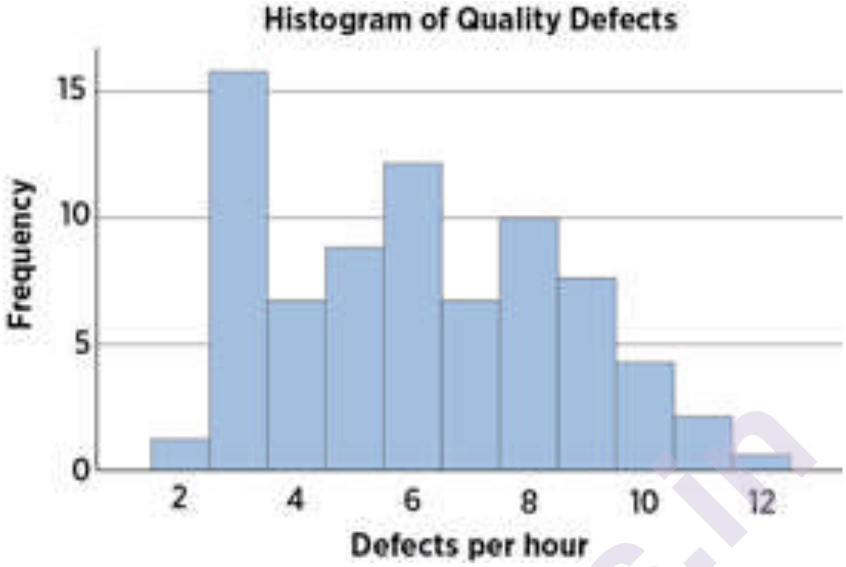
Pareto Diagrams

- ➔ A Pareto chart or diagram, is a specific type of histogram that is based on Pareto's principle, which states that a large number of defects or problems are caused by a small number of causes.
- ➔ Pareto's principle, frequently referred to as the 80/20 rule or 80/20 principle.
- ➔ Which means that eighty percent of the cost of defects are caused by twenty percent of the problems.
- ➔ A Pareto diagram is an ordered bar graph showing the number of defects and their causes ranked by frequency.
- ➔ The bars on the diagram graphically show the number and percentage of causes individually and the line shows the cumulative value.
- ➔ Pareto charts help focus attention on the most critical issues to get the most benefit.



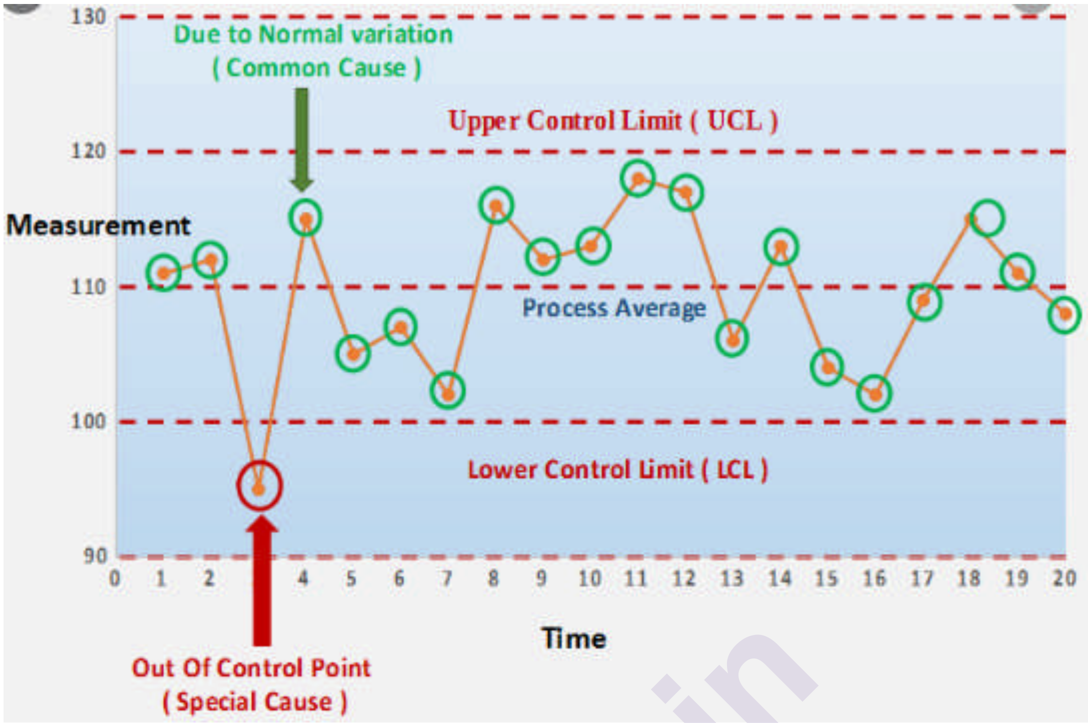
Histograms

- ➔ A histogram is a vertical bar graph that represents the frequency of each measured category (known as bins) of variable.
- ➔ The histogram is particularly useful for identifying common causes.
- ➔ The histogram can be ordered, similar to a Pareto chart, or unordered.



Control Charts

- ➔ Control charts are used to determine if processes are in or out of statistical control. Most processes experience a degree of normal variation i.e. most processes do not achieve target performance all the time.
- ➔ Control charts provide a mechanism for establishing a statistically objective range of acceptable variation around the target performance, thereby enabling attention to be focused on special cause variations (those that fall outside of the established performance limits).
- ➔ Control chart limits are established on the basis of standard deviations from the mean (target) performance. The upper control limit (UCL) and lower control limit (LCL) are established so that 99.73 percent (three standard deviations above and below the mean) of the measured data points fall within the range.
- ➔ The following ones are more common:
- ❑ Rule of Seven, or Seven Run Rule: Seven data points in a row are above or below the mean.
- ❑ Trend of Seven Rule: Seven data points in a row follow a trend up or down.
- ❑ Rule of One: Any single data point is outside of the control limits (upper or lower).

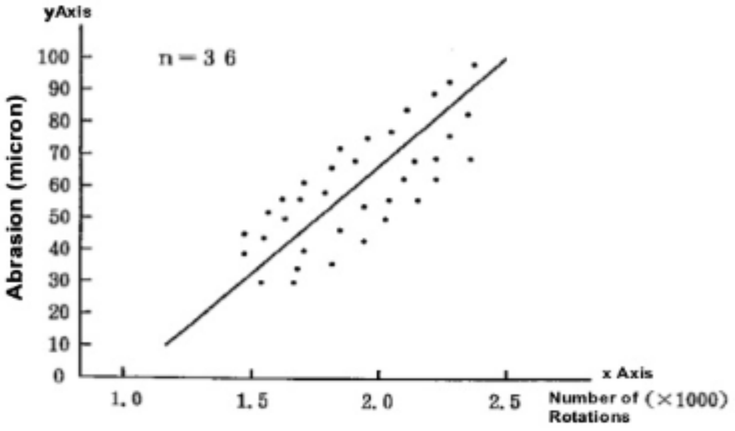


Scatter Diagrams

- ➔ Scatter diagrams plot two variables, the independent variable and the dependent variable, to graphically show the relationship between them.
- ➔ The X-axis in the diagram represents one characteristic (usually the independent variable), and the Y-axis measures the other.
- ➔ To interpret the diagram, look at two characteristics of the clustering:

- ❑ Tightness: The closer the cluster is to a diagonal line drawn through the graph, the more the two variables are likely to be linearly correlated. High correlation between the characteristics means that a change in one characteristic will be accompanied by a change in the other.
- ❑ Direction: If the correlation is positive, then as one variable increases so does the other, and the line will have a positive slope (from lower left to upper right). On the other hand, if the correlation is negative, it implies that as one characteristic increases, the other decreases, and the line will have a negative slope (from lower right to upper left).

Scatter Diagram



7.8 ANSWER THE FOLLOWING :

1. Explain the benefits of the use of SQA standards.

- The ability to make use of the most sophisticated and comprehensive professional methodologies and procedures
- Better understanding and cooperation between users of the same standards:
  - Between team members and between project teams
  - Between software developers and external participants in the project
  - Between suppliers and customers.

2. Describe the contributions made by the use of standards.

- Provision of superior professional methodologies for use in the development process and for its management
- Provision of SQA certification services based on independent professional quality audits
- Provision of tools for "self-assessment" of achievements in planning and operating an organization's SQA system.

3. Describe the general principles underlying quality management according to ISO 9000-3.

- Customer focus – understanding a customer's current and future needs
- Leadership exercised in the creation and maintenance of a positive internal environment in order to achieve the organization's objectives

- Involvement of people at all levels to further organizational goals
- Process approach – activities and related resources perceived and managed as a process
- Systems approach to management – managing processes as a system
- Continual improvement of the organization’s overall performance
- Factual approach to decision-making – decisions based on the analysis of data and information
- Mutually beneficial supplier relationships – emphasis on coordination and cooperation

4. Describe the principles embodied in the Capability Maturity Model (CMM).

- Application of more highly elaborated software quality management methods increases the organization’s capability to control quality and improve software process productivity
- Application of the five levels of the CMM enables the organization to evaluate its achievements and determine what additional efforts are needed to reach the next capability level
- Process areas are generic, with the model defining “what” and leaving the “how” to the implementing organizations, i.e., the choice of life cycle model, design methodology, software development tool, programming language and documentation standard.

❖❖❖❖

8

SOFTWARE RISK MANAGEMENT

Unit Structure

8.0 OBJECTIVES

- Define risk identification and causes , effects , and nature off project risks.
- Apply several analysis techniques that can be used to prioritize and analyze various project risks.
- Describe the various risk strategies
- Describe the risk monitoring and control
- Describe risk evaluation in terms of how the entire risk management process should be evaluated in order to identify best practices.

8.1 INTRODUCTION

Project risk management provides an early warning system for problems that need to be addressed or resolved .Although risk has a certain negative connotation , project stakeholders should be vigilant in identifying opportunities. Although many associate uncertainty with threats, it is important to keep in mind that there is uncertainty when pursuing opportunities, as well.

Plan risk management determines how to approach and plan the project risk management activities. an output of this process is the development of arisk management plan.

Deciding which risks impact the project. Risk identification generally includes many of the project stakeholders and requires an understanding of the project's goal , as well as the project's scope ,schedule, budget , and quality objectives.

Developing procedures and techniques to reduce the threats of risks, while enhancing the likelihood of opportunities.

8.2 RISK ANALYSIS AND MANAGEMENT

Risk analysis and management are series of steps that help a software team to understand and mange uncertainty.

Many problems can plague a software project .A risk is a potential problem –it might happen it might not.

8.2.1 Steps of Risk Analysis and Management

1. Recognizing what can go wrong is the first step , called "risk identification."
2. Each risk is analyzed to determine the likelihood that it will occur and the damage that it will do if it does occur.
3. Risks are ranked, by probability and impact.
4. Finally, a plan is developed to manage those risks with high probability and high impact.

In short, the four steps are

- Risk Identification
- Risk Projection
- Risk Assessment
- Risk Management

Risk always involves two characteristics a set of risk information sheets is produced.

- **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.

8.3 TYPES OF IT PROJECT RISK

What types of risks are we likely to encounter as the software is built??

- **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.
- **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.
- **Business risks** –threaten the viability of the software to be built. Business risks often jeopardize the project or the product. Candidate for top five business risks are

➢ Market risk
➢ Strategic risk
➢ Management risk and
➢ Budget risk

- **Known risks** are those that can be uncovered after careful evaluation of the project plan.
- **Predictable risks** are extrapolated from past project experience (e.g., staffturnover, poor communication with customer, dilution of staff effort as ongoing maintenance requests are serviced).
- **Unpredictable risks –** are the joker in the deck. They can do occur, but they are extremely difficult to identify in advance.

8.4 REACTIVE VS PROACTIVE RISK STRATEGIES

8.4.1 Reactive risk strategies

1. Reactive risk strategies follows that the risks have to be tackled at the time of their occurrence.
2. No precautions are to be taken as per this strategy.
3. They are meant for risks with relatively smaller impact.

8.4.2 Proactive risk strategies

1. Proactive risk strategies follows that the risks have to be identified before start of the project.
2. They have to be analyzed by assessing their probability of occurrence ,their impact after occurrence , steps to be followed for its precaution.
3. They are meant for risks with relatively higher impact.

8.5 RISK IDENTIFICATION

Risk identification is asystematicattemptto 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.

One method for identifying risks is to create a risk item checklist. The checklistcan be used for risk identification and focuses on some subset of known and predictable risks a in the following generic categories:

- **Product size** – risks associated with 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 sophistication of the customer and 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.

8.5.1 Assessing Overall Project Risk

Is the software project we are working on at serious risk?

The questions are ordered by their relative importance to 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 fully in the definition of requirements?
5. Do end –users have realistic expectations?
6. Is the 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 any one of these questions is answered negatively, mitigation, monitoring, and management steps should be instituted without fail.

2.5.2 Risk Components and Drivers

The risk components are defined in the following manner:

- **Performance risk-** the degree of uncertainty that the product will meet its requirements and 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 impact categories– negligible, marginal, critical, or catastrophic.

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 in excess of $ 500K	
	2	Significant degradation to non-achievement of technical performance	Nonresponsive or unsupportable software	Significant financial shortage budget overrun likely	Unachievable 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 value of $ 100K to $500K	
	2	Some reduction in technical performance	Minor delays in software modifications	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 $1K to $100K	
	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 $ 1K	
	2	No reduction in technical performance	Easily supportable software	Possible budget under run	Early achievable IOC

1) The potential consequence of undetected software errors or faults.
2) The potential consequence if the desired outcome is not achieved.

8.6 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. Establishing a scale that reflects the perceived likelihood of a risk.
2. Delineating the consequences of the risk.
3. Estimating the impact of the risk of the project and the product.
4. Noting the overall accuracy of the risk projection so that there will be no misunderstandings.

8.6.1 Developing a Risk Table

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 the planned	PS	70%	2	
End –users resist system	BU	40%	3	
Delivery deadline will be tightened	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	30%	2	
Staff turnover will be high	ST	60%	2	

I. A risk table provides a project manager with a simple technique for risk projection.

II. A sample risk table is illustrated in Figure. The risk table is sorted by probability and impact to rank risks.

III. A project team begins by listing all risks in the first column of the table. This can be accomplished with the help of risk item checklists referenced. Each risk is categorized in the second column (e.g. PS implies a project size risk, BU implies business risk).

IV. 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.

V. Next, the impact of each risk is assessed.

VI. The categories for each of four risk components – performance, support, cost, and schedule- are averaged to determine an overall impact value.

VII. 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 drops drop to bottom.

VIII. This accomplishes first- order risk prioritization. The project manager studies the resultant sorted table and defines a cutoff line. The cutoff line implies that only risks that lie above the line will be given further attention.

IX. Risks that fall below the line are re- evaluated to accomplish second- order prioritization.

X. A risk factor that has a high impact but a very low probability of occurrence should not absorb a significant amount of management time.

XI. All risks that lie above the cut off line must be managed.

XII. The column labeled RMMM contains a pointer into a Risk Mitigation, Monitoring and Management Plan or alternatively, a collection of risk information sheets developed for all risks that lie above the cutoff.

How do we assess the consequences of a risk?

The following steps are recommended to determine the overall consequences of a risk.

I. Determine the average probability of occurrence value for each risk component.

II. Using figure determine the impact for each component based on criteria shown.

III. 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
RE = P * C

Where P is probability of occurrence for a risk, and C is cost to the project should the risk occur.

8.7 RISK ASSESSMENT

At this point in the risk analysis process we have established a set of triples of the form:
$[r_i , l_i , x_i]$

Where r_i is risk, l_i is the likelihood (probability) of the risk, and x_i 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.

Therefore, during risk assessment we perform the following steps:

I. Define the risk referent levels for the project.

II. Attempt to develop a relationship between each (ri, li,xi) and each of the referent levels.

III. Predict the set of referent points that define a region of termination, bounded by a curve or areas of uncertainty.

IV. Try to predict how compound combinations of risks will affect a referent level.

8.8 RISK MITIGATION, MONITORING, AND MANAGEMENT

- An effective strategy must consider three issues:
- ✓ Risk avoidance
- ✓ Risk monitoring
- ✓ Risk management and contingency planning
- High staff turnover in any organization will have a critical impact on project cost and schedule.
- To mitigate the 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 * 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.
- ✓ Assign a backup staff member for every critical technologist.

8.9 THE RMMM PLAN

1. 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.
2. The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan
3. Some software teams do not develop a formal RMMM document. Rather, each risk is documented individually using a risk information sheet(RIS).
4. Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence.

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.

Risk information sheet			
Risk ID : PO 2-4-32	Date: 5/9/02	Prob: 80%	Impact : high
Description : 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.			
Refinement / context : 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 interface 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.			
Mitigation / monitoring: 1. Contact third party to determine conformance with design standards. 2. Press for interface standards completion; consider component structure when deciding on interface protocol. 3. Check to determine number of components in sub condition 3 category; check to determine if language support can be acquired.			
Management / contingency plan / trigger : RE computed to be $ 20,200. Allocate this amount within project contingency cost. Develop revised schedule assuming that 18 additional components will have to be custom built; allocate staff accordingly. Trigger : Mitigation steps unproductive as of 7/ 1 /02			
Current status : 5/12/02: Mitigation steps initiated.			
Originator: D. Gagne		Assigned : B.Laster	

8.10 SUMMARY

Risk identification should include identifying both threats and opportunities. Since most risks are interrelated and can affect the project in different ways, the project stakeholders should take a broad view of project risks.

Risk assessment allows the project stakeholders to determine what risks require a response. The goal of project risk management is not to avoid each and every risk at all costs, but to make well- informed decisions as to which risks are worth taking and which risks require a response. A well informed decision requires an analysis of the probability of a particular risk occurring and its likely impact.

Risk strategies define how the project stakeholders will respond to risk. It include (1) accepting or ignoring the risk (2) avoiding the risk (3) mitigating or reducing the likelihood and / or impact of the risks , and (4) transferring the risk to someone else.

Once the risk response plan has been completed and the project is underway, the various risks identified must be monitored and controlled. This process should include vigilance on the identified and unidentified threats and or opportunities. As these risks present themselves, project risk owners should make resources available and respond to risk in an appropriate manner, as outlined in the risk response plan.

Risk evaluation provides a key to learning and identifying best practices. A formal and documented evaluation of a risk response or episode can help an organization evaluate its entire risk management approach and provide insight for future project teams that may have to deal with a similar risk in the future.

Sample Questions:

1) What is project risk?
2) What is project risk management?
3) What is the purpose of risk analysis and assessment?
4) What is risk monitoring and control?
5) Why can identifying IT project risks be difficult?
6) Discuss the risk strategies?

Reference Books:

1. Software Engineering, 5th and 7th edition, by Roger S Pressman, McGraw Hill publication.
2. Managing Information Technology Project 6thedition , by Kathy Schwalbe , Cengage Learning publication.
3. Software Engineering 3rd edition by KK Agarwal , Yogesh Singh , New Age International publication.
4. Information Technology Project Management by Jack T Marchewka Wiley India publication.
5. Software Engineering for students: A Programming Approach by Douglas Bell, Pearson publication.
6. Software Engineering Project Management by Richard H. Thayer Wiley India Publication.

❖❖❖❖

Module VII

9

PROJECT IMPLEMENTATION PLAN AND CLOSURE

Unit Structure

9.1 OBJECTIVE

After studying this particular chapter, you will be able to understand

- How to prepare project from start to end?
- Understand the activities taken up for implementation.
- Understand the need to carry out project closure.
- Discuss the problems that may arises if proper project closer is not carried out.
- Assess the activities needed to be performed to close a project.

9.1 INTRODUCTION

When we handled a particular project we consider that the project must be perfect and run without any interruption or any error. For running the project without error/mistake detailed step-by-step process required. There are four phases i.e planning, building, implementation and closure determine the aspect of project.

Project implementation is a stage where all actions are takes place which mentioned in planning. In implementation project details and coordination of team is most important. For obtaining the outcome accurate, exact resources also required. The project implementation phase has two basic functions: execution of work and proper delivery In closure all activities are finalized before delivering the project to customer. Closure activity can be done after successful completion of project or due to any interruption in project.

9.2 PROJECT IMPLEMENTATION

9.2.1 What is project Implementation?

- Project Implementation means putting the project into action.
- It will carry out the phases like execution, monitoring, control and production.
- In project implementation decided structures are executed and their analysis done later on.
- The success of the project is determined by proper monitoring and feedback process.

9.2.2 Key factors of successful project implementation

- The plan prepared in planning phase execute with proper action plan to achieve goal.
- To know about each and every step in detail prepare documentation.
- Communication must be carried out to determine the success and failure factors of the project.
- If prepared plan is failed due to some unknown factors, then take quick decision to change it.
- Form a consensus about the changes and implement immediately.

9.2.3 Methods of project implementation

Following are the various methods of project implementation-

1. **Critical Chain Method**

- In this method actual result is compared with expected result.
- If the expected result is not shown, then sometimes plan or goals needs to be changed.
- To implement this, we can use the following tools: -

✓ Kick-off phase
✓ Staff meeting
✓ Problem solving
✓ Review meeting
✓ Risk analysis

2. **Extreme Project Management**

- This method is used to handle complex project or the project which not provide certain output.
- It uses the approach of Extreme Programming where phases are not fixed as well as activities are also not to fixed.
- This implementation focus on human factor rather than phases.

3. **Critical Path Technique**

- It is used to tell that how the project activities are schedule or planned.
- In this path from start to end is setup and if that path is long then analysing will done to understand what are the critical task involve in it.
- If some task might not complete with in given time period, they will come under scrutiny.
- This technique is used for those projects where deadline matters.

4. **Program Evaluation and review technique(PERT)**

- This technique was developed for optimizing efficiency.
- Program evaluation is systematic approach for collecting, analysing and using information for project.
- PERT helps in scheduling and coordinating all task required in project.
- PERT helps in increasing use of resources.
- Long term project used this technique.

5. **Agile Method**

- In this method we identify the requirements and find solution to complete them.
- In this set of principles that are depends upon value centred approach.
- In this method project divided into short sprint with planning and improvement.
- This technique used for multiple iterations.
- The techniques used for agile methods are Scrum, DSDM, Kanban, FDD

6. **Classic Technique**

- This is one of the simplest way of managing and implementing projects.
- This method mentioned the plan for activities held in future and suggest task to complete them.
- It also suggests some resources those can full fill the activity.
- To obtained quality feedback was taken and activities were monitor to achieve deadlines.

9.2.4 Process of project implementation

- For successful project, all the necessary steps are taken as precautions.
- The delivery of the project means giving out a final product of your work.
- The project management includes mapping out the idea, processing the project, analysing the outcomes, discussing the project, and finally executing the outcome.
- Along with this risk mitigation and success probabilities are also checked.
- Before presenting the idea to customer we should know background and views about them because every customer is different.
- Project Implementation Steps are as follows:

1. Have a firm eye on the process and the budget

- In the implementation phase, it is crucial to look at the bigger picture.
- There is various project monitoring software to ensure that the management focuses on the end-goal and is not engulfed in petty, unimportant details.
- It is also essential to respond quickly to changes at every step of execution and have the entire team on-board to implement these.
- Have an explanation in place for overriding costs that spurt in the last stages. These might be to adhere strictly to the schedule or unaccounted inflationary pressures.
- The processes involved in the whole project may exceed costs, or there might be some problems during the management phase. The team may get distracted with unwanted suggestions and may not focus on the project’s main target.
- So supervising the overall activities and keeping track of the happenings is necessary.

2. Prepare Reports along the way

- In the implementation phase, the stakeholders business demand updates and status report always. Being up-to-date with the progress and documenting it neatly comes in handy to present the statistics to the stakeholders at any point in time.
- While you execute the project idea to the clients or stakeholders, you have to be mentally prepared for all the queries raised. They may ask about a minor aspect and small proceedings of any of the preparation phases.
- The project presenter should be ready with all the recorded data during the initial stages, like planning and building up.
- To avoid any trouble during the implementation, start making reports, and collect data from day one. Hence you will be prepared with the data that will act as your backup strategy.

3. Hold periodic meetings regularly

- Looking at the bigger picture, the person might get lost along the way and falls off the track.
- To ensure that everyone is on the same page with the progress, it is essential to hold general and specific meetings within smaller groups. Keep an eye on production numbers and revenue goals in agenda meetings.
- This step is crucial to maintain team spirit. The project is not a one-person job. Instead, several hands are needed to make it successful. Without a team effort, the project would lack strength.
- So regular meetings will also encourage sharing ideas, feedback on ongoing proceedings, and brainstorming sessions.

4. Rekindle enthusiasm in the working personnel

- It is necessary to review progress along the way; it is equally important to successfully reward the staff to complete small and big targets.
- Performance Review is an essential aspect of the Project Implementation phase.
- It accounts for a significant part of enhancing individual productivity that aids the overall performance in the long run.

5. Resolve Issues at the initial stage

- At this crucial stage of project implementation, resolving issues as soon as or before they emerge becomes hugely significant.
- Paying attention to details, loss of team enthusiasm, scope creep, or dip in the quality should be tackled as soon as it begins to appear.
- This early response to stress leads to a smoother implementation.

9.2.5 Approaches of Project Management

1.Waterfall model

- It is traditional approach to project management.
- It uses sequential or linear approach to develop project.
- The project is broken into sequence of task.

2. Agile Approach

- It uses iterative development approach.
- Agile basically means respond to change and it providing changes in all type of organisation.
- This approach is used to provide qualitative product in given time period.

3. Scrum Approach

- It shows the progress in project by making the use of cycles.
- The activities start by sprint master and he remove all obstacles.
- Scrum is a framework that helps teams to work together.

4. Kanban

- It is a popular framework to implement agile and DevOps Software.
- This approach manage work by balancing demands with available capacity.
- It requires full transparency of work.

5. Scrum ban

- It is an agile development methodology which is hybrid of Scrum and Kanban.
- It provides the structure of Scrum with the flexibility and visualization of Kanban.
- It will provide flexibility to adapt to stakeholder and production needs without feeling overburdened by their project methodology.

6. Extreme Programming

- ➢ It is also agile development framework.
- ➢ It produces higher quality software to fulfil customer needs.
- ➢ XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.

7. Adaptive Project Framework

- ➢ It is also known as adaptive Project Management.
- ➢ It is going to accommodate unknown factors that can crop up during the project.
- ➢ It will manage the complexity in case of uncertainty.

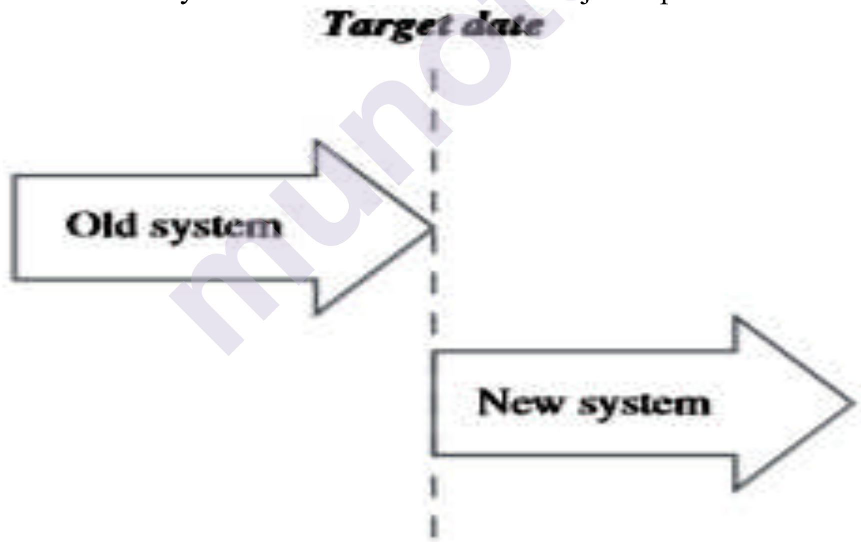
8.Lean methodology

- ➢ It is based on elimination all forms of waste and increase customers perceived.
- ➢ It is a way to optimize people, resources, efforts and energy.
- ➢ It is based on continuous improvement and respect for people.

Along with the above approaches project implementation also having the following three approaches-

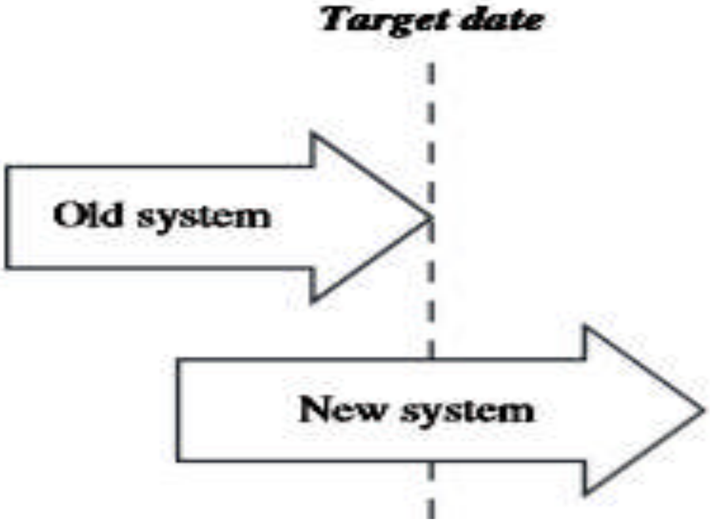
1. **Direct cutover-**

- ✓ In this old system is shut down and new system start.
- ✓ This approach is used when existing system is not performing the functions and require to replace by new system.
- ✓ In this if system fails it does not show major impact on business.



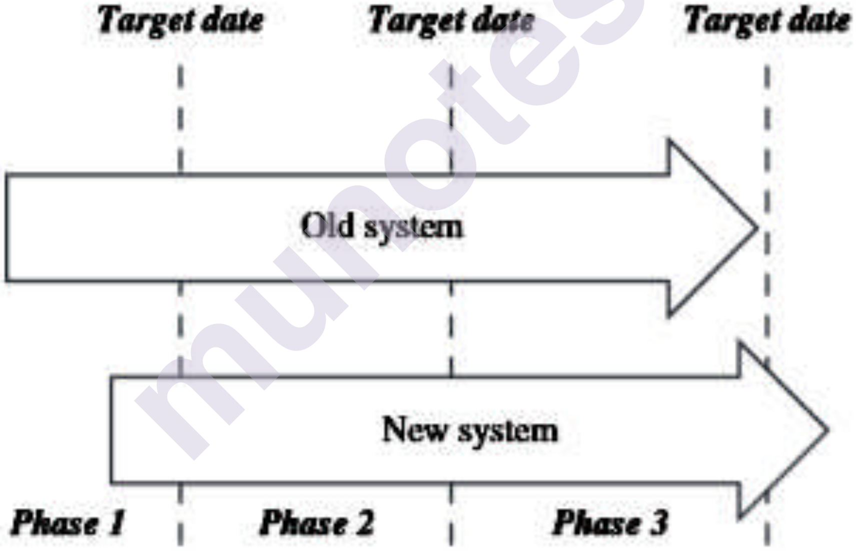
2. **Parallel cutover-**

- ✓ As the name suggest this approach allow old and new system to run concurrently on time.
- ✓ This approach is used when failure of system shows major impact on business.
- ✓ It provides confidence that before relying on new system it works efficiently.



3. **Phased**

- ✓ This approached is used to introduce software system to different areas of the organisation.
- ✓ Example-accounting information system package by first implementing the general ledger component, then accounts payable and accounts receivable, and finally payroll.



9.3 CLOSURE

Every project must come to an end sometime or other. The project manager decides which is the suitable time to close the project. Once it is decided then project manager check that project is close by using proper project closure activities. This is the last phase of project life cycle.

Project closure also involves handover project to customer, passing documentation to business, release staff working on that project, inform stockholders about closure.

Project closure activities can broadly have divided into Administrative Closure and Contract Closure.

Administrative closure insure that all project deliverables are achieved. In contract closure all terms of the contacts with customer are verified to check all requirements are met and satisfactory closed.

In case of multiphase project various closeout activities may have carried out at the end of every phase.

9.3.1 Reasons for Project Closure

There are two main reasons to close the project

1. All the stated goals are successfully completed.
2. Project is unlikely to achieve its stated objectives and has to be prematurely terminated.

Most of the time the manager tries to close the project by satisfying all goals successfully. There can be many reasons for prematurely terminating a project. A few important reasons for project termination are as follows:

1. Lack of resources-

- A project may have terminated due to shortfall of resources.
- Resource shortfall includes change of priority, financial strain on company, bankruptcy of company, change in management of company.

2. Changed business need of customer-

- Customers stated requirements quite change after initiation of the project.
- This make the outcome not useful in that situation
- Example-Customer initially told to prepare inventory management software and later on decides to outsource its entire inventory management activity to third party.

3. The perceived benefits accruing from project no longer remain valid-

- The expected time period of obtaining benefit from project may alter.
- During implementation of the particular software some competitor may launch similar type of software in market.

4. Changes to the regulatory policies-

- External factors such as government policies can have the adverse effect on the project and due to that project needs to be terminated.

5. Key terminology used in the project becoming obsolete during project execution-

- Software is being developed based on core technology.

- Those technologies get outdated before the project completes. Then, there may be reason for terminating project.

6. **Risk become high-**

- Even after project have been carried out throughout the risk, sometimes the risk become so high that project could have adverse effect on the company.
- Impact will be shown such as unsettling its financial soundness, inviting negative publicity, safety hazards.

9.3.2 Why are projects not properly closed?

Here we are considering some of the reasons as why project closure may be delayed or not be properly closed.

1. **Lack of interest by project team-**

- Project closing activity require some creativity, the project team may loss interest to participate in these activities.
- Sometimes project team members already start working on other project so they have no interest in earlier project.
- Team members also be reluctance on part of project which require redeployment after project gets closed.

2. **Emotional factor-**

- After working on the particular project team members and project manager may become emotionally attached to project and they want the project to continue as long as possible.

3. **Indecision regarding project closure-**

- Often some tough decisions are required to be taken by project manager or senior manager to solve the problems arises in the project.
- If they are not able to take the decision on time, then premature termination of the project carried out.

4. **Underestimation of how fast know-how can get lost and how much implicit knowledge exist with team member-**

- Team members working on that project build up significant knowledge on that project.
- It is often underestimated by stakeholders as how much knowledge pertaining to project exist with team members and how fast can the knowledge decay and get lost.

9.3.3 Problems of improper project closure

When the project is not closed in appropriate time period several problems can arise. Some of the important problems that may be faced include the following:

1. **Time and cost overrun-**

- If the time period for the project is delay, then cost runs up expenditure in the meanwhile leading to cost overrun.
- If the time exceed then project shows no value.

2. Locking up valuable human and other resources-

- When project closing is delay then redeployment and other resources gets delayed.
- As a result, valuable resources and manpower that can used in other project gets wasted.

3. Stress on project personnel-

- The person who is appointed on particular project having experience about projects that they deploy.
- The team members will be stressed if they are not doing anything challenging, missing out learning opportunity etc.

9.3.4 Project closure process

To perform the project closure activity, it requires to carry out certain steps. If project facing termination problem, then manager will ask top management and customer that whether they want to continue it or stop it.

The following are the steps that are consider for project closure process-

1. Getting client acceptance-

- Client acceptance mechanism varies project to project.
- If the client is internal, then acceptance tends to informal.
- If the client s external, then acceptance obtained from formal contract process.
- In formal contract acceptance done by the client and sometime written acceptance is also required.

2. Achieving project deliverable-

- Now a day's emails are used to send the documentation so we can easily retrieve the documents.
- Project achieved should be properly documented so that if we required it in future then problems will not arise.
- Document used as deliverable should contain information regarding description of documents, application, stored location, contact information of person.

3. Performing Financial Closure-

- Every project will start only after financial grant is approved.
- Grant may consider the components such as capital and contingency budget.
- Before closure we have to check all payments are completed and have been reconciled.

4. Post implementation project review-

- It is called as post-mortem because it is used to perform critical analysis so that we can improve and avoid mistakes in future.
- Following are some of the steps for conducting post implementation project review-

1. **Conduct project survey-**

- ✓ The main goal of project survey is to collect various types of information regarding project.
- ✓ Questionnaire is design to collect the information. Following points are including in questionnaire-
- ❖ Project performance
- ❖ Administrative performance
- ❖ Organizational structure
- ❖ Team performance
- ❖ Techniques of project management
- ❖ Risk management

2. **Collection of objective information-**

- ✓ It is most critical aspect of project where we study about various metrics.
- ✓ Real data helps to focus discussion on critical issues during post implementation process.
- ✓ The metric which used to collect information are cost metric, schedule metric, quality metric.

3. **Debriefing meeting-**

- ✓ It is preparatory meeting that helps to make final project review meeting focused on most relevant aspects.
- ✓ The main purpose of meeting is to check what was successful and what was not.

4. **Final project review**-

- ✓ This review basically include issues of project planning and tracking, results of various development phase, specification in design and testing.

5. Project closeout report-

- ➢ This report shows the result obtained from various project closeout task.
- ➢ It also contains recommendation for improvement to be used by other project of similar size and scope.
- ➢ It includes the following things-
- ✓ Project description
- ✓ What worked well
- ✓ The factors that impeded the performance of the project.
- ✓ A prescription for other projects to follow.

6. Releasing Staff-

- ➢ This is final step of project closeout process.
- ➢ This step tells about meeting before the team members disperse to different project and recognizing exceptional performance by the team members and recognizing the experience.

9.3.5 Administrative Closure

As we know that project closure activity broadly divided into administrative closure and contact closure.

The definition provided by Gray and Larson following are five circumstances for ending the project-

1. **Normal-**

- ✓ When all the requirements of the project complete then normal closure perform.
- ✓ In normal closure cost, quality and scope are handled properly.

2. **Premature-**

- ✓ Premature closure occurs when project is pushed forcefully in early completion even though not completed its functionality.

3. **Perpetual-**

- ✓ This closure is basically known as runaway projects.
- ✓ This project is result from delay or scope which is not clearly mentioned.

4. **Failed-**

- ✓ Projects are not able to complete its defined requirements, then it failed.
- ✓ Due to failure of project it shows effects on cost factor.

5. **Changed priorities-**

- ✓ Sometime project may have terminated due to some policy changed made in requirements.
- ✓ In this even though the project is never closed officially resources are released and assign to some other project.

As we consider that the project closure must done normally but unfortunately if project not closed normally then project manager and team should prepare for the following reality

- ✓ Team members are concerned about future jobs
- ✓ Bugs still exist
- ✓ Resources are running out
- ✓ Documentation attains paramount importance
- ✓ Promised delivery dates may not be met
- ✓ The players may possess a sense of panic

Summary-

- ❖ Project implementation is the phase where we bring visions and plan in reality.
- ❖ After completing requirement analysis project implementation starts.
- ❖ Project manager conduct project implementation activity.

- The deliverable for your project provide all product and services that you provide to customer.
- Once the information system has been built or purchased, it must be tested adequately in order to make installation of the system go more smoothly.
- Three approaches to implementation were discussed in this chapter. The first approach, called direct cutover, provides the quickest means for implementing the system.
- The parallel and phased approaches are less risky alternatives, although implementation may take longer.
- The phased approach may be appropriate when implementing an upgrade or modular system in different departments or at different geographical locations.
- Once the information system has been implemented, the project manager and team must plan for an orderly end to the project.
- Projects can be terminated for a variety of reasons. If project scope is complete then project complete under normal condition.
- A useful way to gain acceptance is the development of a final project report.
- This report provides a history of the project and outlines how each deliverable was completed and meets the standards of the client or sponsor.
- The report should also address any open items or issues so that they can be completed within a reasonable time.
- This report can serve as a foundation for the project team's final meeting with and presentation to the key stakeholders of the project.
- In this chapter we discussed with various reasons of closing project and problems of improper closure.
- In this chapter we also discussed with process of closure.

Self-Learning Topics: Ethics in Projects, Multicultural Projects

References-

- Information Technology Project Management by Jack T Marchewka Wiley India publication.
- Software Project Management by Bob Hughes Mc GrawHill publication

❖❖❖❖