Introduction:

- The main goal of software project management is to enable a group of developers to work effectively towards the successful completion of a project.
- Project management involves use of a set of techniques and skills to steer a project to success.
- Project manager is usually an experienced member of the team who essentially works as the administrative leader of the team. For small project, single member of team handle both project management and technical management responsibilities.
- The responsibilities of the **technical leader** includes addressing issues such as ***which tools and techniques*** to use in the project, high-level solution to the problem, ***specific algorithms*** to use etc.
- Our focus is to learn why ***management of software projects*** is much **more complex** than managing many other types of projects

Discussion Points

1. Responsibilities and activities of a software project manager.

Management of software projects is much more complex than management of many other types of projects. Main factors contributing to the complexity of managing a software project

1. Invisibility:

Software remains ***invisible***, until its development is complete and it is operational. Anything that is invisible is ***difficult to manage and control***.

Example: house building project

Project manager:

Monitor the **milestones** that **have been completed** by the development team and the **documents** that have been produced—which are rough indicators of the progress achieved.

> ***"Invisibility of software makes it difficult to assess the progress of a project and is a major cause for the omplexity of managing a software project."***

2. Changeability:

Changes to the **business practices**, changes to the hardware or underlying software (e.g. operating system, other applications), or just because the client changes his mind.

"Frequent changes to the ***requirements and the invisibility*** of software are possibly the two major factors making *software project management a complex task*."

3. Complexity:

Even moderate sized software has **millions of parts (functions)** that interact with each other in many ways. Complexity of the functioning of a software product in terms of the basic parts

making up the software, many types of risks are associated with its development This makes managing these projects much more difficult as compared to many other kinds of projects.

4. Uniqueness: Every software project is usually associated with many unique features or situations. This makes every project much different from the others. As a result, a software project manager has to confront many unanticipated issues in almost every project that he manages

5**. Exactness of the solution:** Mechanical components such as **nuts and bolts** typically work satisfactorily as long as they are within a **tolerance of 1 percent** or so of their specified sizes. Parameters of a **function call** in a program are required to be in complete conformity with the function definition. This requirement not only makes it **difficult to get a software product up** and **working,** but also makes **reusing parts** of one software product in another difficult.

6. Team-oriented and intellect-intensive work:

In a software development project, the life cycle activities not only highly ***intellectintensive,*** but each member has to typically **interact, review, and interface** with several other members, constituting another dimension of complexity of software projects.

Mechanical proejcts are (labour intensive).

3.1 RESPONSIBILITIES OF A SOFTWARE PROJECT MANAGER:

Job Responsibilities for Managing Software Projects:

- It is very difficult to objectively describe the precise job responsibilities of a project manager.
- The job responsibilities of a project manager range from invisible activities like building up of team morale to highly visible customer presentations.
- Most managers take the responsibilities for project proposal writing, project cost estimation, scheduling, project staffing, software process tailoring, project monitoring and control, software configuration management, risk management, managerial report writing and presentation, and interfacing with clients.
- These activities are certainly **numerous and varied**. We can still broadly classify these activities into two major types—project planning and project monitoring and control
- **Project planning**: Project planning is undertaken immediately after the **feasibility study phase** and before the starting of the **requirements analysis and specification phase**.
- **Project planning** involves **estimating several characteristics of a projec**t and then planning the project activities based on these estimates made.
- The initial project plans are revised from time to time as the project Progresses and more project data become available.
- **Project monitoring and control**: Project monitoring and control activities are undertaken once the development activities start. The focus of project monitoring and control activities is to ensure that the software development proceeds as per plan.

Skills Necessary for Managing Software Projects:

1. ***Theoretical knowledge*** of various project management techniques
2. Good qualitative judgment and decision taking capabilities.
3. Latest software project management techniques such as ***cost estimation, risk management, and configuration management,*** etc.,
4. Good communication skills and the ability to get work done.
5. Tracking and controlling the ***progress of the project***, customer interaction, managerial presentations, and team building.
6. Three skills that are most critical to successful project management are the following:
   - Knowledge of project management techniques.
   - Decision taking capabilities.
   - Previous experience in managing similar projects

3.2 Project Plannig:

Once a project has been found to be feasible, software project managers undertake **project planning.**
Project planning is undertaken and completed before any development activity starts

During project planning, the project manager performs the following activities.
1. Estimation: The following project attributes are estimated.
• **Cost:** How much is it going to **cost to** develop the software product?
• **Duration:** How long is it going to take to develop the product?
• **Effort:** How much effort would be necessary to develop the product?
2. Scheduling: After all the necessary project parameters have been estimated, the schedules for manpower and other resources are developed.
3. Staffing: Staff organisation and staffing plans are made.
4. Risk management: This includes risk identification, analysis, and abatement planning.
5. Miscellaneous plans: This includes making several other plans such as **quality assurance plan**, and configuration management plan, etc.
Figure 3.1 shows the order in which the planning activities are undertaken. Observe that size estimation is the first activity that a project manager undertakes during project planning. Size is the most fundamental parameter based on which all other estimations and project plans are made.

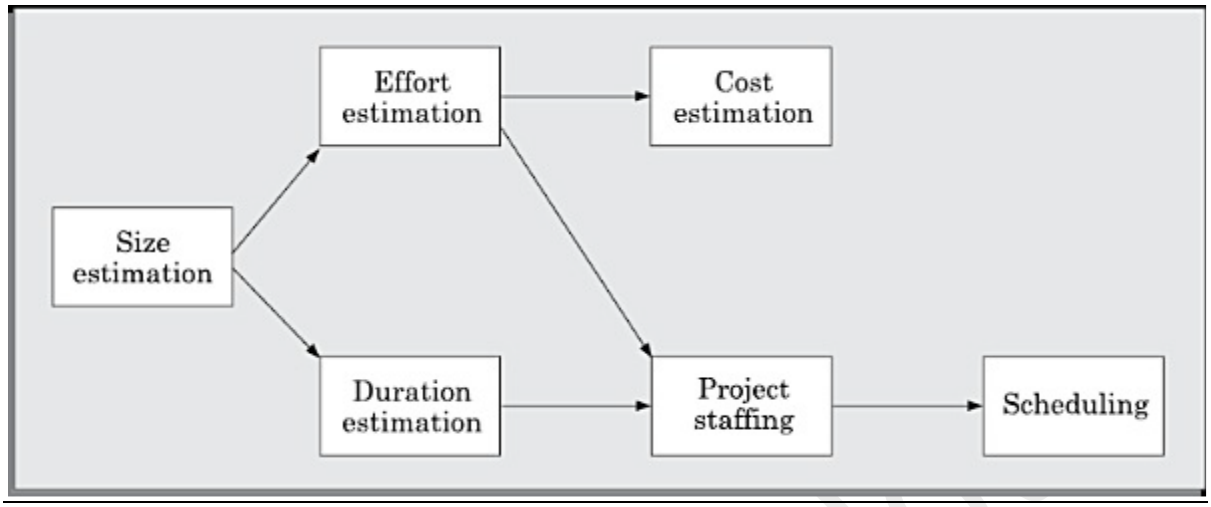


Figure 3.1: Precedence ordering among planning activities.

3.2.1 Sliding Window Planning

- It is usually very difficult to make **accurate plans** for **large projects** at Project initiation. In the **sliding window planning technique**, starting with an **initial plan**, the project **is planned more accurately** over a **number of stages.**
- At the start of a project, the project manager has **incomplete knowledge** about the nitty-gritty of the project.
- His information base gradually improves as the project progresses through different development phases. The **Complexities of different project activities** become clear, some of the **anticipated risks get resol**ved, and new risks appear. The project parameters are **re-estimated periodically** as understanding grows and also aperiodically as **project parameters change.**
- By taking these developments into account, the project manager can plan the subsequent activities more accurately and with increasing levels of confidence

3.2.2 The SPMP Document of Project Planning

Once project planning is complete, project managers document their plans in a **software project management plan (SPMP) document**

1. **Introduction**: (a) Objectives (b) Major Functions (c) Performance Issues (d) Management and Technical Constraints
2. **Project estimates** (a) Historical Data Used (b) Estimation Techniques Used (c) Effort, Resource, Cost, and Project Duration Estimates
3. **Schedule:** (a) Work Breakdown Structure (b) Task Network Representation (c) Gantt Chart Representation (d) PERT Chart Representation
4. **Project resources** (a) People (b) Hardware and Software (c) Special Resources
5. **Staff organisation** (a) Team Structure (b) Management Reporting
6. **Risk management plan** (a) Risk Analysis (b) Risk Identification (c) Risk Estimation (d) Risk Abatement Procedures
7. **Project tracking and control plan** (a) Metrics to be tracked (b) Tracking plan (c) Control plan

8. **Miscellaneous plans:** (a) Process Tailoring (b) Quality Assurance Plan (c) Configuration Management Plan (d) Validation and Verification (e) System Testing Plan (f) Delivery, Installation, and Maintenance Plan

3.3. Project Scheduling: the project tasks is an important project planning activity

The scheduling problem, in essence, consists of deciding which tasks would be taken up when and by whom.

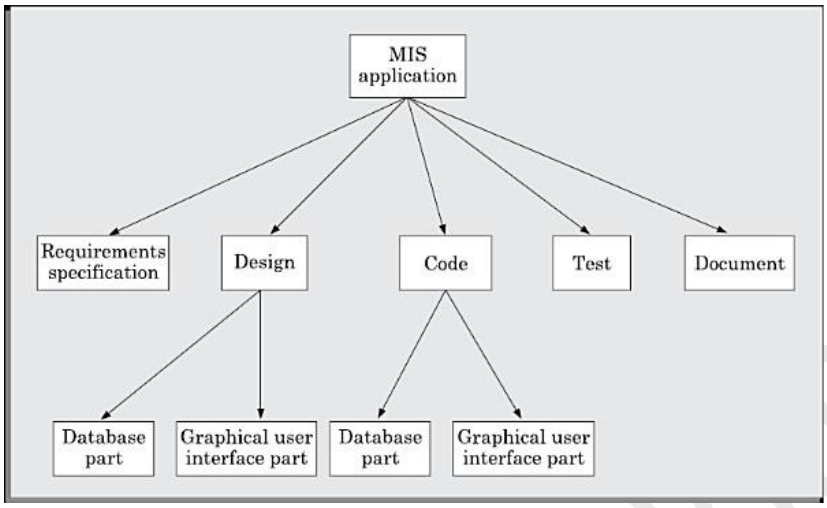
Schedule has been worked ***project manager*** monitors the **timely completion of the tasks** and takes any **corrective action** that may be necessary whenever there is a chance of schedule **slippage.**

Schedule the project activities; a software project manager needs to do the following:

1. **Identify** all the **major activities** that need to be carried out to complete the project.[**Good knowledge of project + development process**]
2. **Break down** each activity into tasks. [Break down structure technique]
3. Determine the **dependency among different tasks. [partial ordering among tasks]**
4. Establish the **estimates for the time durations** necessary to complete the tasks.
5. Represent the information in the form of an **activity network.**
6. **Determine task starting and ending dates** from the information represented in the **activity network.**
7. Determine **the critical path**. A critical path is a **chain of tasks** that determines the **duration of the project**.
8. Allocate **resources to tasks. [Resource allocation is typically done using a Gantt chart]**
9. Project evaluation and review technique **(PERT) chart representation [helps in project monitoring**
and control.]

Good knowledge of the intricacies of **the project** and the **development process** helps the managers to
Effectively identify the important activities of the project **development process.**
Activities are broken down into a **logical set of smaller activities (subactivities). Smallest subactivities are called tasks which are assigned to different developers.**

1. Work Breakdown Structure: Tasks are the lowest level work activities in a WBS hierarchy. They also form the basic units of work that are allocated to the developer and scheduled.



WBS of management information system (MIS) problem

Activities are broken down into a **logical set of smaller activities (subactivities**). The smallest subactivities are called **task**s which are assigned to different developers. WBS provides a **notation** for representing the **activities, sub-activities, and tasks** needed to be carried out in order to solve a problem. Each of these is represented using a rectangle (see Figure 3.7).

The **root of the tree** is labelled by the **project name**. Each node of the tree is broken down into **smaller activities** that are made the children of the node.
How long to decompose?

The decomposition of the activities is carried out until any of the following **is satisfied:**

- A leaf-level **subactivity (a task)** requires approximately **two weeks to develop**.
- **Hidden complexities are exposed**, so that the job to be done is understood and can be assigned as a unit of work to one of the developers.
- Opportunities **for reuse of existing software** components is identified.

Breaking down tasks to too coarse levels versus too fine levels:

Coarse level decomposition:

When the granularity [**execution time**] of the tasks is several months, by the time a problem (schedule \delay) is noticed and corrective actions are initiated, it may be too late for the project to recover.

Fine level:

- On the other hand, if the tasks are decomposed into **very small granularity** (one or two days each), then the milestones get too closely spaced.
- This would require **frequent monitoring of the project status** and entail **frequent revisions** to the plan document.

- This becomes a **high overhead** on the project manager and his **effectiveness in project monitoring and control gets reduced.**

Note: manager needs to **break large tasks into smaller ones**, expecting to **find more parallelism**. However, it is not useful to subdivide tasks into units which take less than a **week or two** to execute.

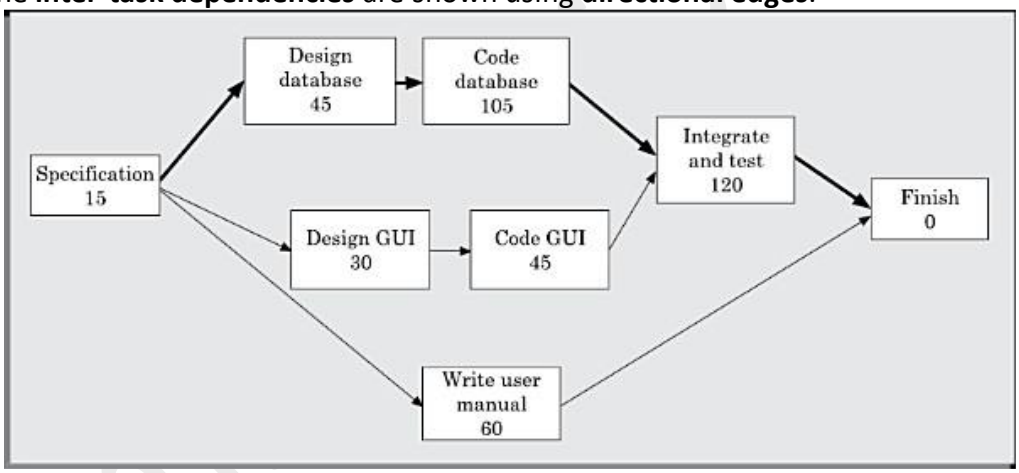
2. Activity Networks:

An activity network shows the **different 1. activities making up a project**, their 2. **estimated durations**, and their 3. **interdependencies**.

Two equivalent **representations for activity networks** are possible and are in use:

Activity on Node (AoN): In this representation,

- Each activity is represented by a **rectangular** (some use circular) node and the **duration of the activity** is shown **alongside** each task in the node.
- The **inter-task dependencies** are shown using **directional edges**.



Activity network representation of the MIS problem.

Activity on Edge (AoE): In this representation,

- **tasks** are associated with the **edges**.
- The edges are also annotated with the **task duration**.
- The **nodes in the graph** represent **project milestones.**

Exercise: Determine the **Activity network representation** for the MIS development project. Assume that the manager has determined the tasks to be represented from the **work breakdown structure** and has determined the **durations and dependencies** for each task as shown in Table.

Task Number	Task	Duration	Dependent on Tasks
T1	Specification	15	–
T2	Design database	45	T 1
T3	Design GUI	30	T 1
T4	Code database	105	T 2
T5	Code GUI part	45	T 3
T6	Integrate and test	120	T 4 and T 5
T7	Write user manual	60	T 1

Project Parameters Computed from Activity Network

3. Critical Path Method (CPM)

A critical path is a **chain of tasks** that determines the **duration of the project**. CPM and PERT are operation research techniques that were developed in the late 1950s.

A critical path consists of a **set of dependent tasks** that need to be **performed in a sequence** and which together take the **longest time to complete**.

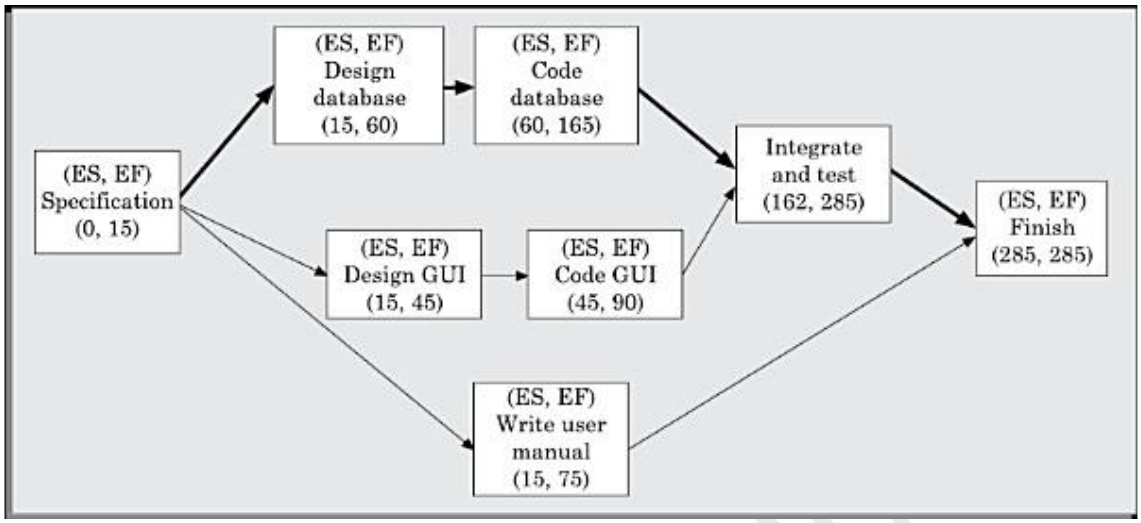
Critical task- has zero slack time

A. CPM is an **algorithmic approach** to determine the **critical paths and slack times** for tasks not on the critical paths involves calculating the following quantities:

1. **Minimum time (MT):** minimum time required to complete the project.
2. **Earliest start (ES):**It is the time of a **task** is the **maximum of all paths from the start to this task**.

   ES for task= ES of the previous task + the duration of the preceding task.
3. **Latest start time (LST**): difference between MT and the maximum of all paths from this task to the finish.
4. **Earliest finish time (EF)**: The EF for a task is the **sum of the earliest start time** of the task and the **duration of the** task.

   EF for the task= Earliest start time of the task + **Duration of the** task
5. **Latest finish (LF):** LF indicates the **latest time** by which a **task can finish** without affecting the final completion time of the project.
6. **Slack time (ST):** The slack **time (or float time**) is the total time that a **task may be delayed** before it will affect the end time of the project



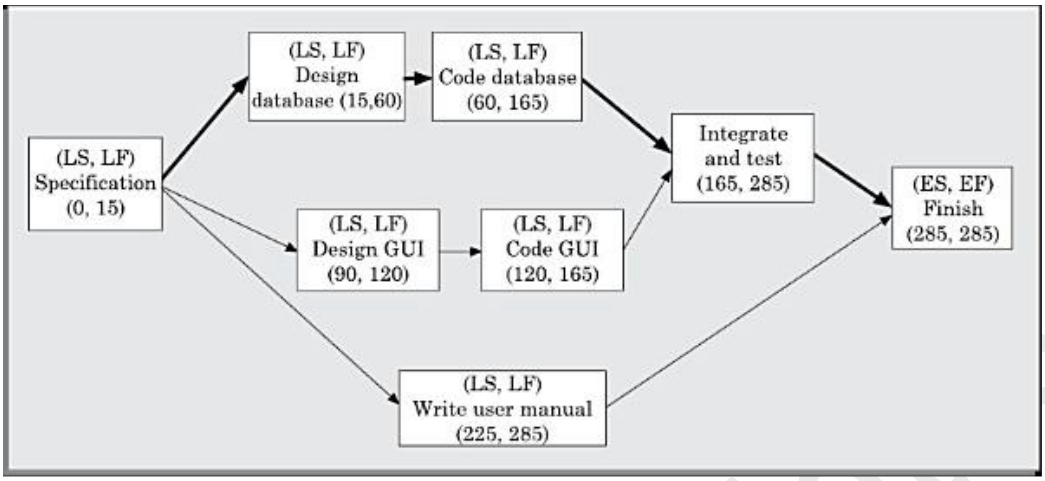
AoN for MIS problem with (ES,EF).

ES(Specification)= **ES of the previous task (Null)** 0 +Duration of preciding task(Null) = 0 + 0 = 0
ES(Design database) = **ES (**Specification) +Duration(Specification) = 0 + 15 = 15
ES(Code database) = **ES (**Design database) +Duration(Design database) = 15 + 45 = 60

EF(Specification **)= Earliest start time** of the task(Specification) + **Duration of the** task(Specification)= 0+15=15
EF(Design database)= ES(Design database) + Duration(Design database) = 15 + 45 = 60
EF(Integrate and test) = ES(Integrate and test) + Duration(Integrate and test) = 165 + 120 = 285

The project parameters for different tasks for the MIS problem can be computed as follows:

1. Compute **ES and EF for each task**.
   Use the rule: **ES = largest EF the immediate predecessors**
2. Compute **LS and LF for each task**.
   Use the rule: **LF = smallest LS of the immediate successors**
3. Compute ST for each task.
   Use the rule: **ST=LF-EF**



AoN of MIS problem with (LS,LF)

LS= Minimum time (MT)- maximum of all paths from this task to the finish.
MT= 285

1. LS(specification) = 285- [Max{15+45+105+120+0},{15+30+45+120+0},{15+60+0}]
   = 285- [Max{285},{210},{75}]=285-285=0
2. LS(Design database)= 285- (45+105+120+0)= 285-270= 15
3. LS(Code Database)= 285- (105+120+0)= 285-225= 60

LF= Smallest LS of the immediate successors

1. **LF(Specification)= 15**
2. **LF(Design Database)=60**
3. **LF(Design GUI)=120**
4. **LF(Write user manual)= 285**
5. **LF(Code Design)= 165**

Table 3.8: Project Parameters Computed From Activity Network

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

Project Parameters Computed From Activity Network

1. ST(Specification)= LF(Specification)-EF(Specification)= 15-15=0
2. ST(Design database)= LF(Design database)-EF(Design database)= 60- 60=0

critical paths are all the paths whose duration equals MT

PERT Charts:

- The activity durations computed using an activity network are only **estimated duration**. It is therefore not possible to estimate the worst case (**pessimistic**) and best case (**optimistic**) estimations using an activity diagram.
- CPM can be used to determine the **duration of a project,** but does not provide any **indication of the probability of meeting that schedule.**
- Project evaluation and review technique **(PERT) charts** are a more sophisticated form of **activity chart.**
- **The** duration assigned to tasks by the project manager are after all **only estimates.**
- PERT charts can be used to determine the **probabilistic times** for reaching various project mile
  stones, including the final mile stone.
- PERT charts like **activity networks** consist of a **network of boxes and arrows**. The boxes represent **activities** and the arrows represent **task dependencies.**
- The duration of an activity is a random variable with some probability distribution.
- PERT charts can be used to determine the **probabilistic times** for reaching various **project mile stones**, including the final mile stone.
- A PERT chart represents the **statistical variations** in the project estimates assuming these to be normal
  distribution.
- PERT allows determining the probability for achieving project milestones based on the probability of completing each task.
- Each task is annotated with three estimates:
  1. Optimistic (O): **The best possible case task completion time.**
  2. Most likely estimate (M): **Most likely task completion time.**
  3. Worst case (W): **The worst possible case task completion time.**
- The three estimates are used to compute the expected value of the standard deviation.
- It can be shown that the entire distribution lies between the interval (M – 3 <= ST) and (M + 3 >= ST), where ST is the standard deviation. From this it is possible to show that— The standard deviation for a task
- ST = (P – O)/6.
- The mean estimated time is calculated as ET = (O + 4M + W)/6.

  Optimistic Time : 20days
  Most likely time: 30 days
  Worst Case(W): 60days

  PERT Weighted average/mean = 20+4*30+60/6= 33.3

  Example :

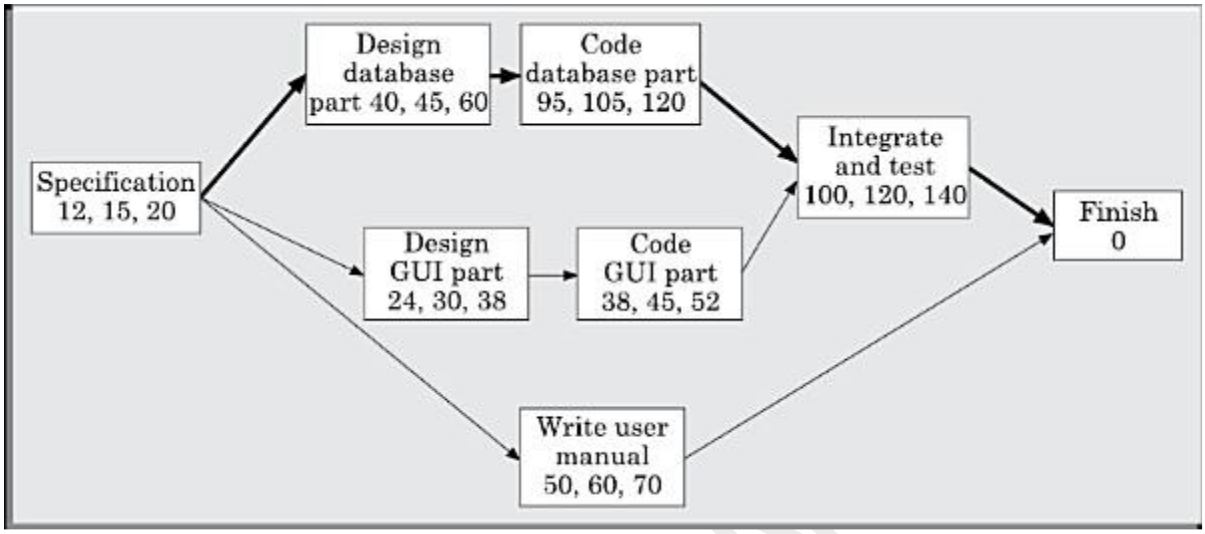


Figure 3.11: PERT chart representation of the MIS problem.

Activity	Predecessor activity	Optimistic(O)	Most likely (M)	Worst Case(W)
Specification (A)	-	12	15	20
Design Database part(B)	A	40	45	60
Design GUI Part(C)	A	24	30	38
Write User Manual(D)	A	50	60	70
Code Database part(E)	B	95	105	120
Code GUI Part(F)	C	38	45	52
Integrate and Test(G)	E,F	100	120	140
Finish(H)	D,G	0	0	0

Expected Time= A = 12+4*15+20/6=15.33
B= 46.66
C= 30.33
D=60
E=105.83
F=45
G=120

3.4 Risk Management:

- A risk is any **anticipated (Predicated) unfavourable event** or **circumstance** that can occur while a project is **underway.**
- If a risk **becomes real**, it can adversely affect the **project and hamper** the successful and timely completion of the project.
- Therefore, it is necessary for the **project manager** to **anticipate/predict** and **identify different risks** that a project is **susceptible t**o, so that contingency (provision for future event) plans can be prepared beforehand to contain each risk.
- Risk management aims at **reducing the chances** of a **risk** becoming **real** as well as reducing the impact of risks that **becomes real.**
- Risk management consists of three essential activities—
    - risk ***identification***
    - risk ***assessment***
    - risk ***mitigation***.

1. Risk Identification

- The project manager needs to anticipate the risks in a project as early as possible.
- As soon as a risk is identified, effective risk management plans are made, so that the possible impacts of the risks is minimised.
- Example: project manager might be worried whether the (Risk 1) **vendors whom you have asked to develop certain modules might not complete their work in time, whether they would turn in poor quality work**,
- (Risk 2) Some of your key personnel might **leave the organisation**, etc. All such risks that are likely to affect a project must be identified and listed.
- There are three main categories of risks which can affect a software project:
    - Project risks
    - Technical risks
    - Business risks

1. Project Risks:

- Contain budgetary, schedule, personnel, resource, and customer-related problems
- Proejct Risk: schedule slippage
- Reason: software is intangible, it is very difficult to monitor and control a software project. The **invisibility** of the product being developed is an important reason of software projects suffer from the risk of **schedule slippage**.

2. Technical Risks:
   - Concern potential **design, implementation, interfacing, testing, and maintenance** problems.
   - Include **ambiguous specification, incomplete specification, changing specification, technical uncertainty, and technical obsolescence.**
   - It occurs due the development team's **insufficient knowledge about the product**.
3. Business risks:
   - Includes the risk of building an **excellent product** that no one wants, losing **budgetary commitments,** etc.

Example:Project: mobile product Development Company

1. What if the project cost escalates and overshoots what was estimated**?: Project risk.**
2. What if the mobile phones that are developed become too bulky in size to conveniently carry?: **Business risk.**
3. What if it is later found out that the level of radiation coming from the phones is harmful to human being?: **Business risk.**
4. What if call hand-off mobile becomes too difficult to implement?: **Technical risk.**

2. Risk Assessment:

- The objective of risk assessment is to **rank the risks in** terms of their **damage causing potential.**
- For risk assessment, first each risk should be rated in two ways:
  - The likelihood of a **risk becoming real (r).[Risk Probability]**
  - The **consequence of the problems** associated with that **risk (s) [Impact]**
- Priority of each risk can be computed as follows: p = r * s
  - Where p is the priority with which the risk must be handled
  - r is the probability of the risk becoming real
  - s is the severity of damage caused due to the risk becoming real

3. Risk Mitigation:

- After all the identified risks of a project have been **assessed, plans** are made to contain the **most damaging and the most likely risks first.**
- There are three main strategies for risk containment[***Action of keeping something harmful under the control or within limits]***
  - Avoid the risk
  - Transfer the risk
  - Risk reduction

1. **Avoid the risk:**
   Risks often arise due to **project constraints (cost, time, scope, quality)** and can be avoided by suitably **modifying the constraints.**

- The different **categories of constraints** that usually give rise to risks are:
  1. **Process-related risk**: These risks arise due to aggressive work schedule, budget, and resource utilisation.
  2. **Product-related risks**: These risks arise due to commitment to challenging product features (e.g. response time of one second, etc.), quality, reliability etc.

3. **Technology-related risks**: These risks arise due to commitment to use **certain technology** (e.g., satellite communication).

- **Examples of risk avoidance**
  1. Discussing with the customer to change the requirements to reduce the scope of the work
  2. Giving incentives to the developers to avoid the risk of manpower turnover, etc.

2. Transfer the risk:

- Involves getting the risky components developed by a **third party, buying insurance cover, etc**.

3. Risk reduction:

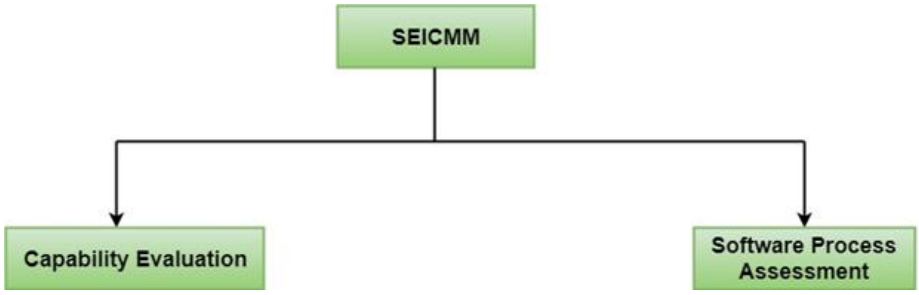
- This involves planning ways to contain the **damage due to a risk.**
- **Risk**: some key personnel might leave **Plan**: new recruitment may be planned**.**
- There can be several **strategies to cope up with a risk**:The project manager must consider
  the **cost of handling the risk** and the **corresponding reduction of risk.**
- **Risk leverage** of the different risks is computed.

$$\text{risk leverage} = \frac{\text{risk exposure before reduction} - \text{risk exposure after reduction}}{\text{cost of reduction}}$$

3.5 SEI capability maturity model:[SEI CMM]

- The Capability Maturity Model (CMM) is a procedure used to develop and refine an **organization's software development process.**
- The model defines a five-level evolutionary stage of increasingly organized and consistently more mature processes.
- CMM was developed and is promoted by the **Software Engineering Institute (SEI)**, a research and development center.
- SEI CMM was originally developed to assist the U.S. Department of Defense (DoD) in software acquisition.
- CMM is a reference model for apprising(telling) the software process maturity into different levels. This can be used to predict the most likely outcome to be expected from the next project that the organisation undertakes.
- SEI CMM can be used in two ways—

Methods of SEICMM



- Capability evaluation provides a way to **assess the software process capability** of an organisation.
- You can use a process-capability study to **assess the ability of a process to meet specifications.**
- Capability evaluation is administered by the **contract awarding authority.**
- Software process assessment is used by an organisation with the **objective to improve its own process capability.**
- **SEI CMM** classifies software development industries into the following five maturity levels:

1. Level 1: Initial

- A software development organisation at this level is characterised by **ad hoc activities.**
- Very **few or no processes** are defined and followed.
- Since software production **processes are not defined**, different engineers follow their own process and as a result development efforts become chaotic (Confusing). Therefore, it is also called chaotic level.
- The **success of projects** depends on **individual efforts and heroics**.
- When a developer leaves the organisation, the successor would have **great difficulty in understanding** the process that was followed and the work completed.
- **No formal project management** practices are followed.
- As a result, **time pressure** builds up towards the end of **the delivery time**, as a result short-cuts are tried out leading to **low quality products.**

2. Level 2: Repeatable:

- At this level, the **basic project management** practices such as **tracking cost and schedule** are established.
- **Configuration management tools** are used on **items identified for configuration control.**
- Size and cost estimation techniques such as function point analysis, COCOMO, etc.,
- There is only **rough understanding** among the developers about the **process being** followed.
- Process is not documented.
- **Configuration management practices** are used for all project deliverables.

- Since the products are **very similar,** the success story on **development of one product** can repeated for another.

3. Level 3: Defined:

- At this level, the processes for both **management and development activities** are defined and documented.
- There is a common organisation-wide understanding of **activities, roles, and responsibilities.**
- The processes though defined, the **process and product qualities are not measured**.
- At this level, the organisation builds up the **capabilities of its employees through periodic training programs.**
- **Review techniques** are **emphasized and documented** to achieve phase containment of errors.
- ISO 9000 aims at achieving this level.

4. Level 4: Managed

- At this level, the focus is on **software metrics**. Both **process and product metrics** are collected
- Quantitative **quality goals** are set for the products. And at the time of completion of development it was checked whether the quantitative quality goals for the product are met.
- Various tools like **Pareto charts, fishbone diagram**s, etc. are used to measure the product and process quality.
- The **process metrics** are used to check if a **project performed satisfactorily**. Thus, the results of process measurements are used to evaluate project performance rather than improve the process.

Level 5: Optimising

- At this stage, **process and product metrics** are collected.
- Process and product measurement data are **analysed for continuous process improvement**.
- **For example**, if from an analysis of the process measurement results, it is found that the **code reviews are not very** effective and a large number of errors are detected only during the unit testing, then the process would be fine tuned to make the review more effective.
- Also, the lessons learned from **specific projects** are incorporated **into the process**.
- Continuous process improvement is achieved both by **carefully analysing the quantitative feedback** from the process measurements and also from application of innovative ideasand technologies.
- At CMM level 5, an organisation would **identify the best software engineering practices** an **innovations** (which may be tools, methods, or processes) and would transfer these organisation wide.
- Level 5 organisations usually **have a department** whose sole responsibility is to assimilate latest tools and technologies and propagate them organisation-wide.

- Since the process changes continuously, it becomes necessary to effectively manage a changing process.
- Therefore, level 5 organisations **use configuration management techniques** to manage **process changes.**

- Except for level 1**, each maturity level** is characterised by several **key process areas (KPAs)** that indicate the areas an organisation should focus to **improve its software process** to this level from the previous level.
- Each of the focus areas identifies a **number of key practices or activities** that need to be implemented.
- In other words, **KPAs capture the focus areas of a level**. The focus of each level and the corresponding key process areas are shown in the Table.

Table 11.1 Focus areas of CMM levels and Key Process Areas

CMM Level	*Focus*	*Key Process Areas (KPAs)*
Initial	Competent people	
Repeatable	Project management	Software project planning Software configuration management
Defined	Definition of processes	Process definition Training program Peer reviews
Managed	Product and process quality	Quantitative process metrics Software quality management
Optimising	Continuous process improvement	Defect prevention Process change management Technology change management

CMM Shortcomings: CMM does suffer from several shortcomings. The important among these are the following:

1. The most frequent complaint by organisations while trying out the **CMM-based process improvement initiative** is that **they understand what is needed to be improved**, but ***they need more guidance about How to improve it.***
2. Another shortcoming (that is common to ISO 9000) is that **thicker documents**, **more detailed information, and longer meetings** are considered to be better. This is in contrast to the principles of software economics—**reducing complexity** and keeping the **documentation to the minimum** without sacrificing the relevant details.
3. **Getting an accurate measure of an organisation's current maturity level** is also an issue. The CMM takes **an activity-based approach** to **measuring maturity**; if you do **the prescribed set of activities** then you are at a **certain level**. There is nothing that characterises or quantifies whether you do these activities well enough to deliver the intended results.

3.6 Project Staffing

ORGANISATION AND TEAM STRUCTURES:

Organisation Structure: three broad ways in which a software development organisation can be structured—

A. Functional format: Development staffs are divided based on the specific functional group to which they belong to.
B. Project format: The development staffs are divided based on the project for which they work
C. In a matrix organization: The pool of functional specialists are assigned to different projects as needed

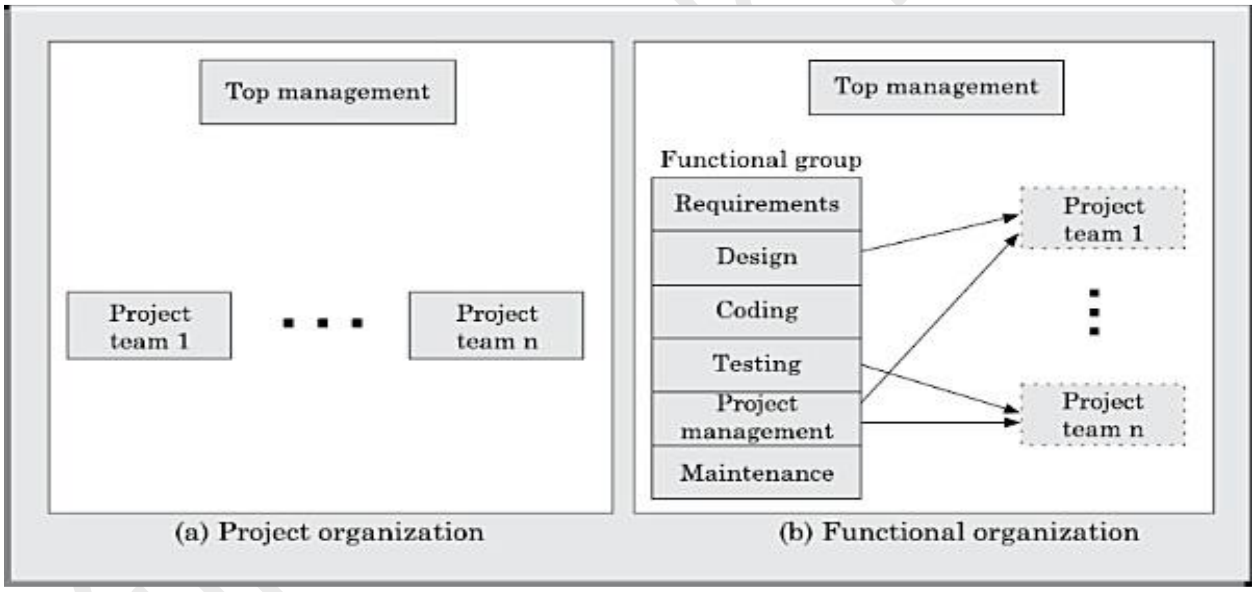


Figure: Schematic representation of the functional and project organisation.

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

Figure: Matrix organisation

Team Structure:

Three formal team structures—

1. chief programmer
2. Democratic
3. mixed control team Organizations

1. Chief programmer team:

- A senior engineer provides the technical leadership and is designated the chief programmer.
- Partitions the task into many smaller tasks and assigns them to the team members.

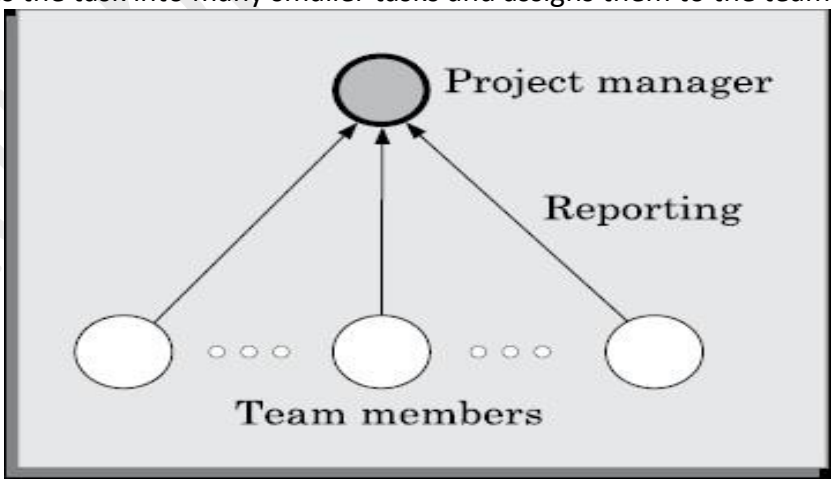
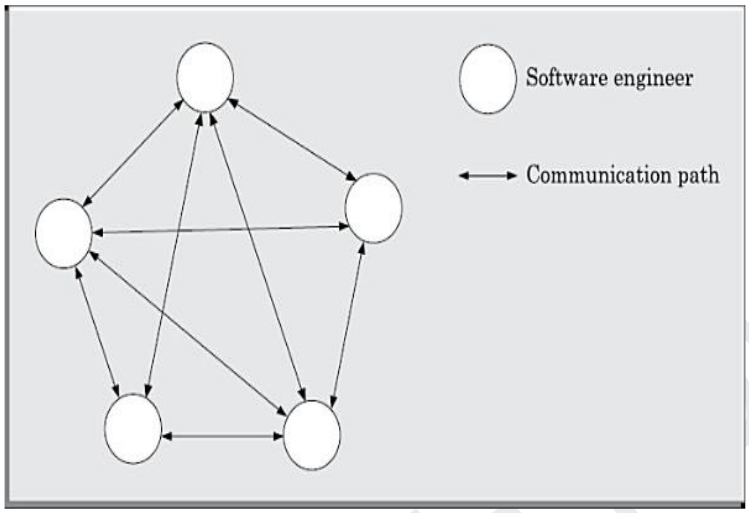


Fig. Chief programmer team structure

2. Democratic team

- **Does not enforce any formal team hierarchy**

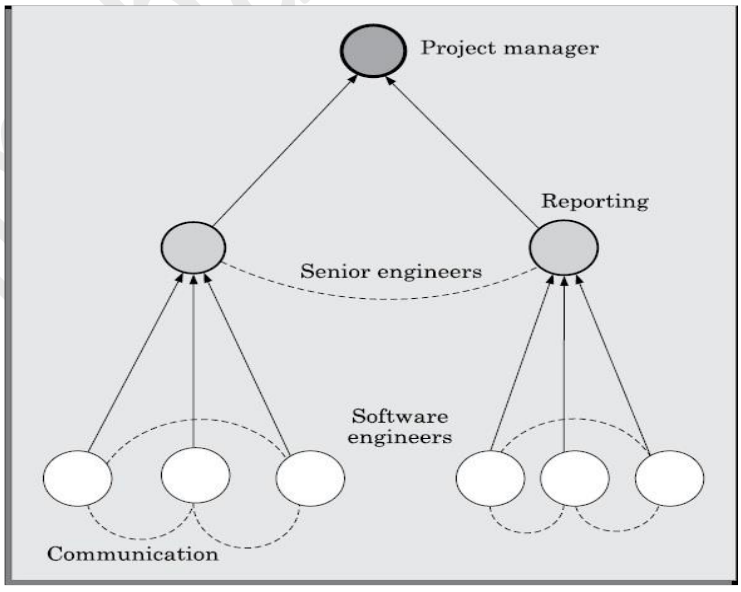
- Manager provides the administrative leadership.
- At different times, different members of the group provide technical leadership.



3. Mixed control team organisation

- Both the ***democratic organisation and the chief programmer organisation.***

Communication paths are shown as dashed lines and the reporting structure is shown using solid arrows



Mixed team structure

- Software project managers usually take the responsibility of choosing their team.
- A common assumption held by managers that **one software engineer is as productive as another**
- worst and the best software developers in a scale of 1 to 30.
- choosing good software developers is **crucial** to the success of a project.

Who is a good software engineer?

- Exposure to systematic techniques, i.e. familiarity with software engineering principles.
- Good technical knowledge of the project areas (Domain knowledge)
- Good programming abilities.
- Good communication skills. These skills comprise of oral, written, and interpersonal skills.
- High motivation.
- Sound knowledge of fundamentals of computer science
- Intelligence.
- Ability to work in a team. Discipline, etc.