Unit - III Software Project Management

📘 Project Scheduling in Software Engineering

Definition: Project Scheduling is the process of **planning tasks**, **allocating resources**, and **setting timelines** to ensure a software project is **completed on time and within budget**.

It defines:

- **What tasks need to be done**
- **When they need to be done**
- **Who will do them**
- **How long they will take**

🎯 Basic Principles of Project Scheduling

1. **Compartmentalization** – Divide the project into smaller manageable tasks or modules.
2. **Interdependency** – Determine relationships between tasks (what depends on what).
3. **Time Allocation** – Estimate the time required for each task.
4. **Effort Validation** – Ensure enough resources are available for each task.
5. **Defined Responsibilities** – Assign responsibilities to team members.
6. **Milestones and Deadlines** – Mark key dates and review points.
7. **Monitoring** – Track progress and update the schedule regularly.

🔧 Project Scheduling Techniques

🔷 1. CPM (Critical Path Method)

🔹 **Definition:** CPM is a technique used to find the **longest sequence of dependent tasks** (called the **critical path**) that determines the **minimum project duration**.

🔹 **Key Concepts:**

- Tasks are represented as **nodes** or **activities**
- Each task has a **fixed duration**
- **No uncertainty** in task time estimates

🔹 **CPM Steps:**

1. List all tasks and their durations
2. Identify dependencies between tasks
3. Draw a network diagram
4. Calculate the **critical path**
5. Highlight tasks with **zero slack** (delays in these will delay the project)

✅ **Advantages:**

- Helps identify most important (critical) tasks
- Easy to use for **well-defined projects**
- Optimizes **time and resource allocation**

❌ **Disadvantages:**

- Assumes **fixed time estimates**
- Less effective for **uncertain or risky** projects

🔷 **2. PERT (Program Evaluation and Review Technique)**

🔹 **Definition:** PERT is used to handle **uncertainty in project scheduling** by using **probabilistic time estimates**.

🔹 **Key Concepts:**

- Each task has **three time estimates**:
  - **Optimistic time (O)** – best-case
  - **Most likely time (M)** – expected case
  - **Pessimistic time (P)** – worst-case

🔹 **PERT Steps:**

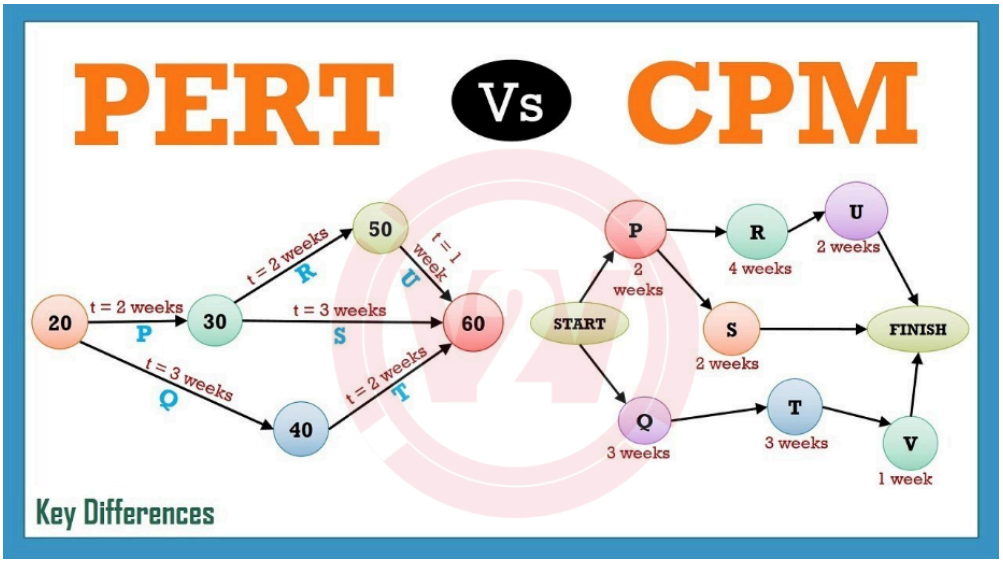
1. List all activities and estimate O, M, P times
2. Calculate Expected Time (TE) for each activity
3. Build the network diagram
4. Find the **critical path** based on TE values

✅ **Advantages:**

- Useful for **complex or uncertain projects**
- Better time estimation using **probabilities**
- Helps in **risk analysis**

❌ **Disadvantages:**

- Requires accurate estimates for O, M, P
- More complex than CPM


PERT
Vs
CPM
20
30
40
50
60
t = 2 weeks
P
t = 3 weeks
Q
t = 2 weeks
R
t = 1 week
U
t = 3 weeks
S
t = 2 weeks
T
START
P
2 weeks
R
4 weeks
U
2 weeks
S
2 weeks
Q
3 weeks
T
3 weeks
V
1 week
FINISH
Key Differences

Difference Between PERT and CPM

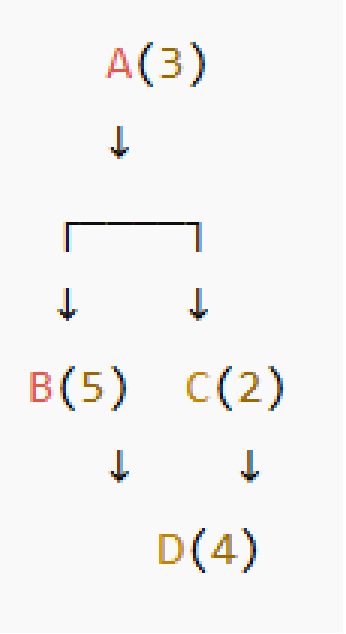
Aspect	PERT (Program Evaluation and Review Technique)	CPM (Critical Path Method)
Focus	Time (scheduling under uncertainty)	Time and cost (for predictable projects)
Nature	**Probabilistic** – used when task durations are uncertain	**Deterministic** – used when task durations are known
Time Estimates	Uses **three estimates:** Optimistic, Most Likely, Pessimistic	Uses **one fixed time** estimate per activity
Application	Used in **R&D, new product development**	Used in **construction, engineering, software**
Critical Path	Helps in identifying uncertain critical paths	Identifies a fixed critical path
Cost Consideration	Does **not focus on cost** directly	Includes **cost analysis** and optimization

📘 1. CPM Example – Critical Path Method

- **Given Tasks:**

Activity	Duration (days)	Predecessor
A	3	-
B	5	A
C	2	A
D	4	B, C

🔹 **Step 1: Draw the Network Diagram**



🔹 **Step 2: Identify All Paths & Durations**

1. **Path 1**: A → B → D = 3 + 5 + 4 = **12 days**
2. **Path 2**: A → C → D = 3 + 2 + 4 = **9 days**

🔹 **Step 3: Find the Critical Path**

- **Critical Path = A → B → D**
- **Minimum project duration = 12 days**

✅ **Why?** It's the **longest path** with **no slack**.

✅ **Final CPM Summary:**

- **Critical Path**: A → B → D
- **Project Duration**: 12 days

📘 **2. PERT Example – Program Evaluation and Review Technique**

🔹 **Given Activity with Time Estimates:**

Activity	Optimistic (O)	Most Likely (M)	Pessimistic (P)
A	2	4	6

🔹 Step 1: Use PERT Formula to Calculate Expected Time

$$TE = \frac{O + 4M + P}{6}$$

$$TE = \frac{2 + 4(4) + 6}{6} = \frac{2 + 16 + 6}{6} = \frac{24}{6} = \boxed{4 \text{ days}}$$

🔹 Step 2: Calculate Variance (optional for risk analysis)

$$\text{Variance} = \left(\frac{P - O}{6}\right)^2 = \left(\frac{6 - 2}{6}\right)^2 = \left(\frac{4}{6}\right)^2 = \boxed{0.44}$$

✅ **Final PERT Summary:**

- **Expected Time (TE)**: 4 days
- **Variance**: 0.44 (used for risk and confidence analysis)

📘 **Project Tracking in Software Engineering**

🔷 **Definition: Project tracking** is the process of **monitoring and controlling** the software project's progress to ensure it stays on **schedule, within budget, and meets quality** expectations.

It involves:

- Tracking task completion
- Identifying delays
- Updating progress regularly
- Making adjustments if needed

🎯 **Purpose of Project Tracking:**

- Compare **planned vs actual** progress
- Identify and fix **bottlenecks**
- Ensure **resource efficiency**
- Keep the team and stakeholders **informed**

📊 1. Timeline Chart

🔹 **Definition:** A **Timeline Chart** shows the **sequence of events or tasks** in a software project over time. It is a **simple horizontal bar** representation of the schedule.

🔹 **Characteristics:**

- Time is shown on the **horizontal axis**
- Tasks/milestones are placed in sequence
- Good for showing **major events and deadlines**

✅ **Advantages:**

- Easy to understand
- Great for **high-level planning**
- Visualizes the **order of tasks**

❌ **Disadvantages:**

- Does not show task dependencies
- Not suitable for **complex projects**

✅ 1. Timeline Chart – Example

📘 **Scenario:**

You are developing a **student management system**.

🔹 **Major Project Events (with Dates):**

Milestone	Planned Date
Requirements Gathering	Day 1
Design Completion	Day 4
Coding Start	Day 6
Testing Start	Day 10
Deployment	Day 14

📊 Timeline Chart (Text Representation)

```
Day →       1     4     6     10    14
            ↓     ↓     ↓     ↓     ↓
Event →  [Req] [Design] [Code] [Test] [Deploy]
```

—

📊

2.

Gantt Chart

🔹 **Definition:** A **Gantt Chart** is a detailed bar chart that shows:

- **Start and end dates** of tasks
- **Duration** of each task
- **Dependencies** between tasks
- **Progress tracking** (completion percentage)

It is the **most widely used tool for project tracking**.

🔹 **Components of Gantt Chart:**

- Vertical axis: **Tasks/Activities**
- Horizontal axis: **Time scale (days/weeks/months)**
- Bars: Represent the **duration of each task**
- Arrows/Lines: Show **dependencies** between tasks
- Progress shading: Shows **% of task completed**

✅ **Advantages:**

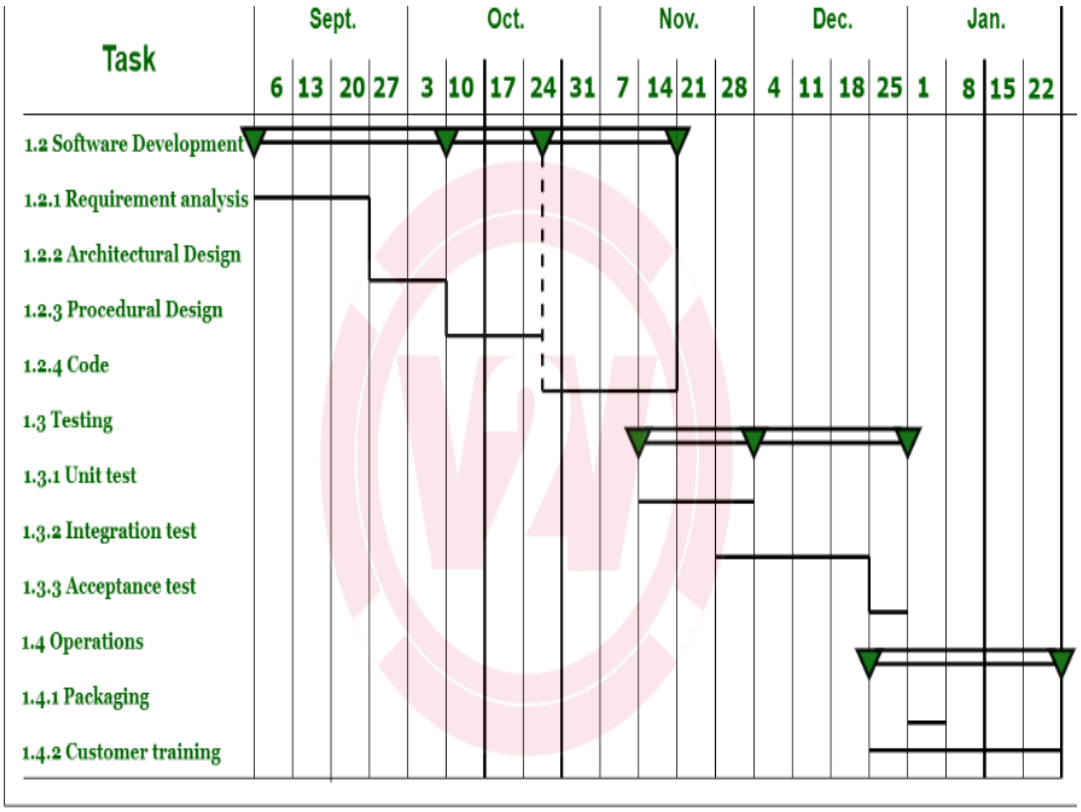
- Clearly shows **project schedule and status**
- Tracks **progress visually**

- Helps with **resource and time management**

❌ **Disadvantages:**

- Can become **complex for large projects**
- Needs to be **updated regularly**

📊 **Gantt Chart (Example)**



Gantt Chart