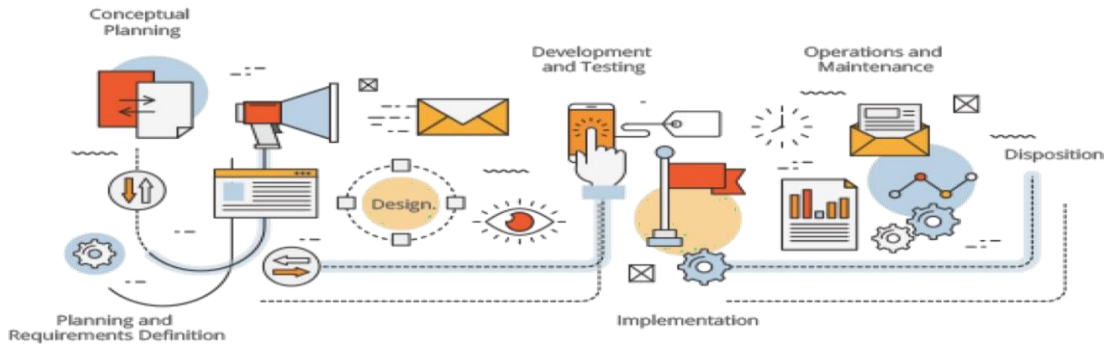
Software Engineering

Project Management



So, why project management?

Because without it, chaos wins.

Project management = Turning ideas into outcomes, without the stress and mess.

This one’s a big deal in tech because software projects are *notoriously* tricky:

- changing requirements,
- tight deadlines,
- complex codebases, and
- lots of people with different roles.

Software Project Management

- Software & Project Management
  - Corporate America spends more than $275 billion each year on approximately 200,000 application software development projects
  - Most of these projects will fail for lack of skilled project management
- Management problems were more frequently dominant cause than technical problems
- Schedule overruns were more common (89%) than cost overruns (62%)

KPGM's Survey in UK

Why It’s Essential:

1. Manages Changing Requirements
In software, clients *always* want changes — sometimes mid-sprint. Project management (especially Agile or Scrum) allows you to adapt while staying on track.

2. Coordinates Cross-Functional Teams
Dev, QA, UX, product, marketing — all have different goals. A project manager makes sure they’re aligned and working together, not pulling in different directions.

3. Keeps Deadlines Realistic
Software takes time. SPM helps scope the work properly, break it into sprints/phases, and avoid the “we'll just add this one more feature” trap that causes delays.

4. Tracks Progress and Quality
With tools like Jira, Trello, or Azure DevOps, you can monitor tasks, bugs, and velocity — making sure things are both functional and clean (no spaghetti code!).

5. Manages Risk
Will a third-party API break? Will a dev leave mid-project? Is the design scalable? SPM identifies risks early and builds plans to handle them.

6. Improves Communication
Regular stand-ups, retros, demos — all baked into Agile/PM practices to make sure no one's in the dark and feedback loops stay tight.

7. Ensures Better Product Delivery
End goal: deliver software that actually meets user needs and performs well. That's the whole reason SPM exists.

So, when exactly is project management performed?

Throughout the **entire lifecycle of a project**. But it happens in phases, with specific responsibilities at each stage.

Project management is not a one-time thing — it's an ongoing process that kicks off before the first line of code is written, and continues until well after delivery.

🧩 1. **Initiation Phase** – *"Should we do this?"*

- **Project management starts here.**
- Define the problem or opportunity.
- Set goals, scope, and stakeholders.
- Do a feasibility check or cost-benefit analysis.
- Result: Project charter or proposal.

📋 2. **Planning Phase** – *"How do we do it?"*

- Probably **the most PM-heavy phase.**
- Set timelines, budgets, resource needs, and deliverables.
- Break down work (WBS – Work Breakdown Structure).
- Identify risks and mitigation strategies.
- Choose a methodology (Agile, Waterfall, etc.).

🚧 3. **Execution Phase** – *"Let's build it."*

- Project manager monitors team progress.
- Keeps communication flowing.
- Resolves blockers or issues.
- Manages scope changes and makes sure deliverables align with the plan.

📊 4. **Monitoring & Controlling** – *"Are we on track?"*

- Happens **in parallel with execution.**
- Track KPIs: Time, cost, scope, quality.
- Handle deviations from the plan.
- Manage change requests and updates.

✅ 5. **Closing Phase** – *"We're done... right?"*

- Deliver final product.
- Confirm acceptance with stakeholders.
- Conduct retrospectives or lessons learned.
- Archive documents, release resources, and close contracts.

Software Project Management (2)

Success Factors

1. User involvement – 20 points
2. Executive Support – 15 points
3. Clear Business Objectives – 15 points
4. Experienced Project Manager – 15 points
5. Small milestones – 10 points
6. Firm basic requirements – 5 points
7. Competent staff – 5 points
8. Proper planning – 5 points
9. Ownership – 5 points
10. Others – 5 points

Most of these points are of Management concern

Software Project Management (3)

Primary causes of software runaway

- Project Objectives not fully specified
- Bad planning and estimating
- Technology new to the organization
- Inadequate/No project management methodology
- Insufficient senior staff on the team
- Poor performance by Supplier of hardware/software

In a nutshell

"Organizations that attempt to put software engineering discipline in place before putting project management discipline in place are domed to fail"

In short, the statement emphasizes that organizations focusing solely on software engineering discipline without implementing project management discipline are likely to face challenges and potential failure in their projects. Effective project management ensures **coordination**, **clarity of goals, resource management, risk mitigation, and stakeholder communication**, all of which are critical for project success. Balancing both disciplines is key to maximizing project success in software development endeavors.

Project Management Life Cycle

Processes/Activities of the **5 phases (Process Groups 2004)** are as:

1. Scope the Project

- State the Problem/ Opportunity
- Establish the project goal
- Define project Objectives
- Identify the success criteria
- List assumptions, risks, obstacles

2. Develop detail Plan

- Identify project activities
- Estimate activity duration
- Determine resource requirements
- Construct / analyze project network
- Prepare project proposal

3. Launch the Plan

- Recruit and organize project team
- Establish team operating rules
- Level resources
- Schedule work packages
- Document work packages

4. Monitor/ Control Progress

- Establish progress reporting system
- Install change control tools/process
- Define problem-escalation process
- Monitor project progress vs plan
- Revise Plan

5. Close Out

- Obtain client acceptance
- Install project deliverables
- Complete project documentation
- Complete post-implementation audit
- Issue final project report

Knowledge Areas for Project Management

1. **Project Integration Management**
2. **Project Scope management**
3. **Project Time Management**
4. **Project Cost Management**
5. **Project Quality Management**
6. **Project Human Resource Management**
7. **Project Communications Management**
8. **Project Risk Management**
9. **Project Procurement Management**

[PMBOK PMI]

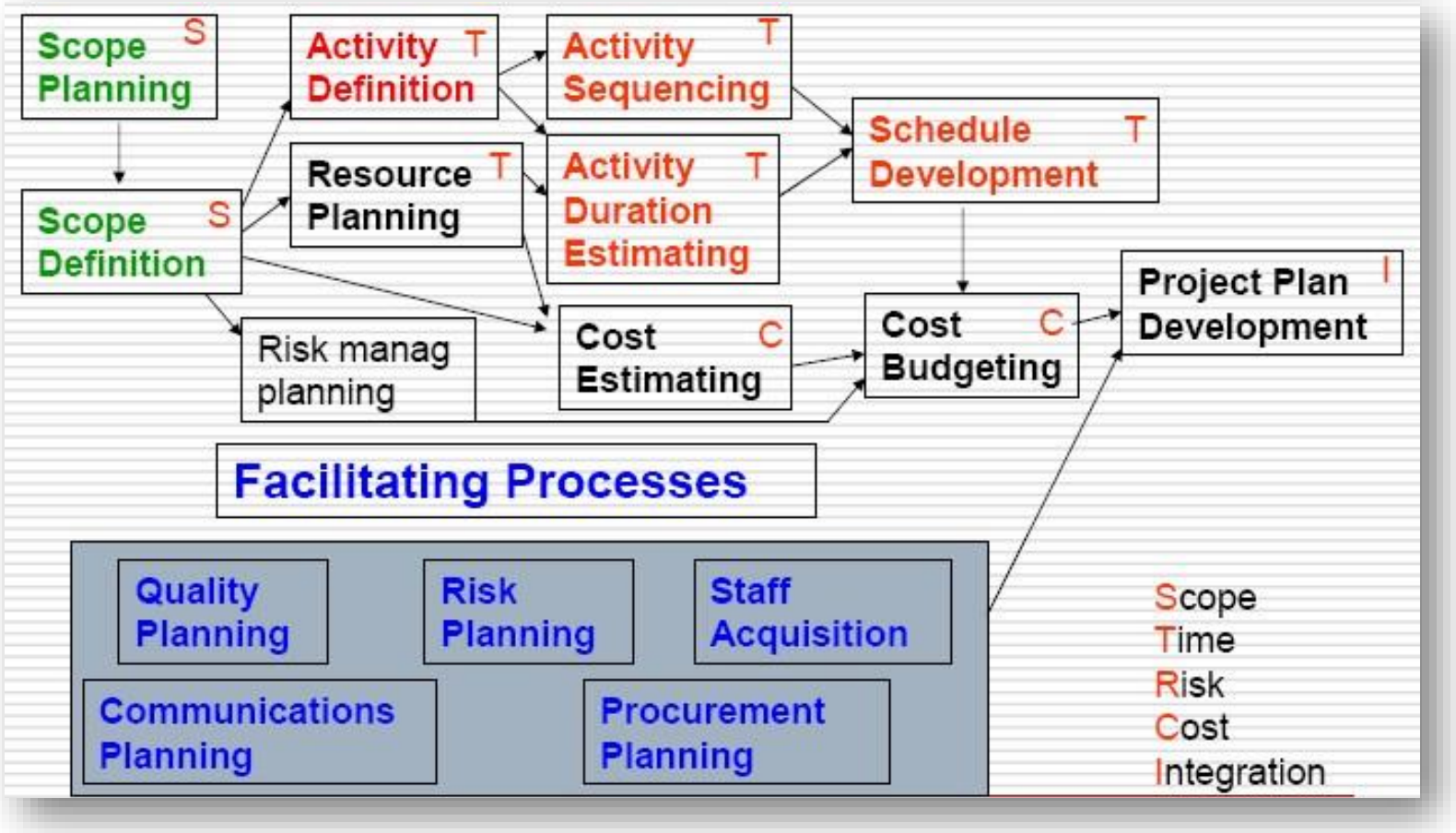
1. Software Project Planning

- **The purpose of the Project Plan is to define and establish the management strategy for achieving the goals of the project.**

The project development plan is used to:

- **Guide execution**
  Helps the team know what to do and when to do it.
- **Document assumptions**
  Makes clear any conditions or guesses made during planning.
- **Record decisions**
  Keeps track of why certain choices were made (e.g., tech stack, features).
- **Enable communication**
  Ensures everyone (developers, testers, clients) stays on the same page.
- **Define management reviews**
  Sets times to check progress and make sure goals are being met.
- **Provide a baseline**
  Acts as the "measuring stick" for tracking actual progress vs. the plan.

Planning Process Flow



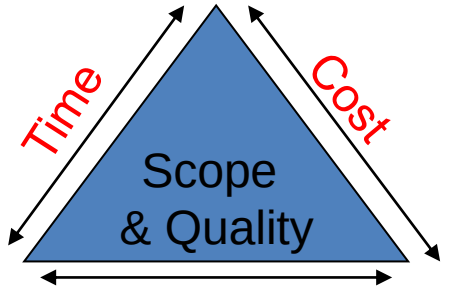
Project planning process

1. **Initiation**: Define project scope, objectives, and key stakeholders.
2. **Planning**: Detail requirements, create a work breakdown structure, develop schedules, estimate resources, identify risks, and establish communication and quality plans.
3. **Execution**: Implement the project plan, manage resources, communicate progress, and ensure quality.
4. **Monitoring and Controlling**: Track progress, manage risks, address deviations, and maintain compliance.
5. **Closure**: Complete deliverables, assess lessons learned, finalize documentation, release resources, and evaluate project success.

2. Scope Definition

The first step - define the Scope

- Projects are dynamic systems that must be kept in equilibrium
- Not easier at all, as shown from the dynamics of the situation
- Area inside the triangle (Scope & Quality) is bound by the Lines (Cost, Time, Resource)



"Scope" **The term may refer to:**

a) **Product scope - the features and functions that are to be included in a product or service.**
b) **Project scope - the work that must be done in order to deliver a product (with the specified features and functions).**

1. ***Product Scope*** **is defined in the product requirements and is the subject of the product life cycle**
2. ***Project Scope*** **is defined in the project charter and is the subject of the project plan**

(Kathy-ch4)

Decomposition

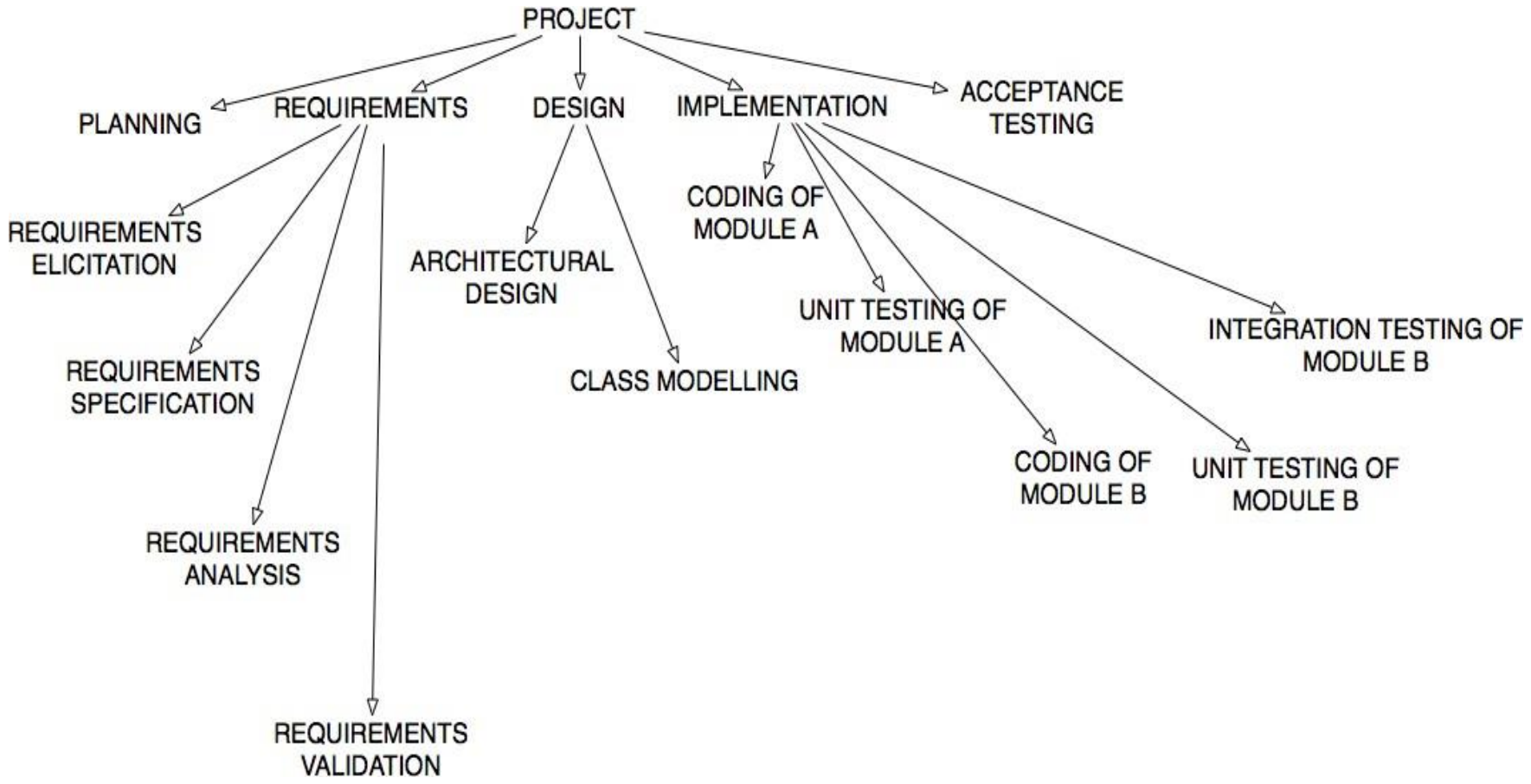
Subdividing the major project deliverables into smaller, more manageable Components (products) in sufficient detail to support future project activities (planning, executing, controlling, and closing); requires Steps:

1) **Identify the major elements (deliverable) of the project.**
2) **Identify constituent elements (Work Products) of the deliverable.**
3) **Decide if adequate cost and duration estimates can be developed at this level of detail for each element.**

1. Identify the major elements of the project.

- **In general, the major elements will be the project deliverables and project management product. For example:**
  - **The phases of the project life cycle may be used as the first level of decomposition with the project deliverables repeated at the second level.**
  - **The organizing principle within each branch of the WBS may vary.**

Project Analysis by Work Breakdown Structures (WBS)



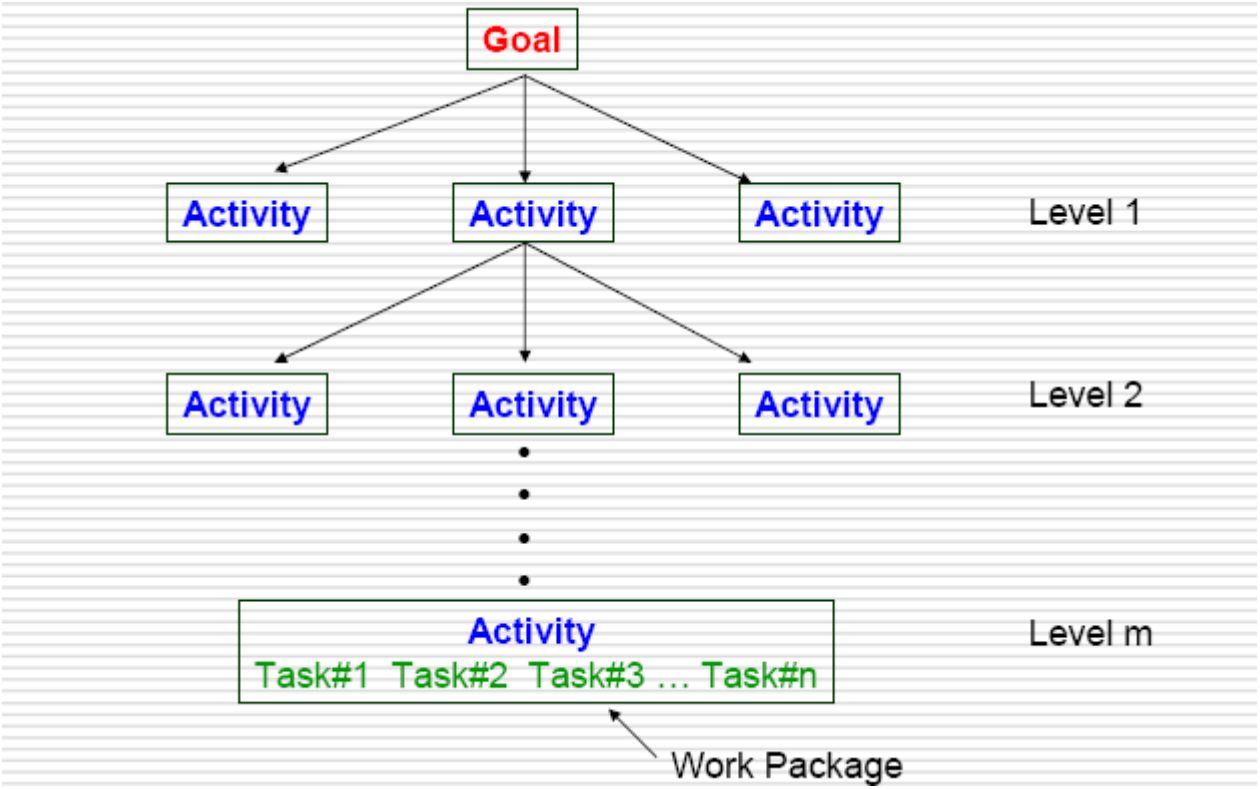
Purpose of Creating WBS

Improve accuracy of cost, time, and resource estimates.

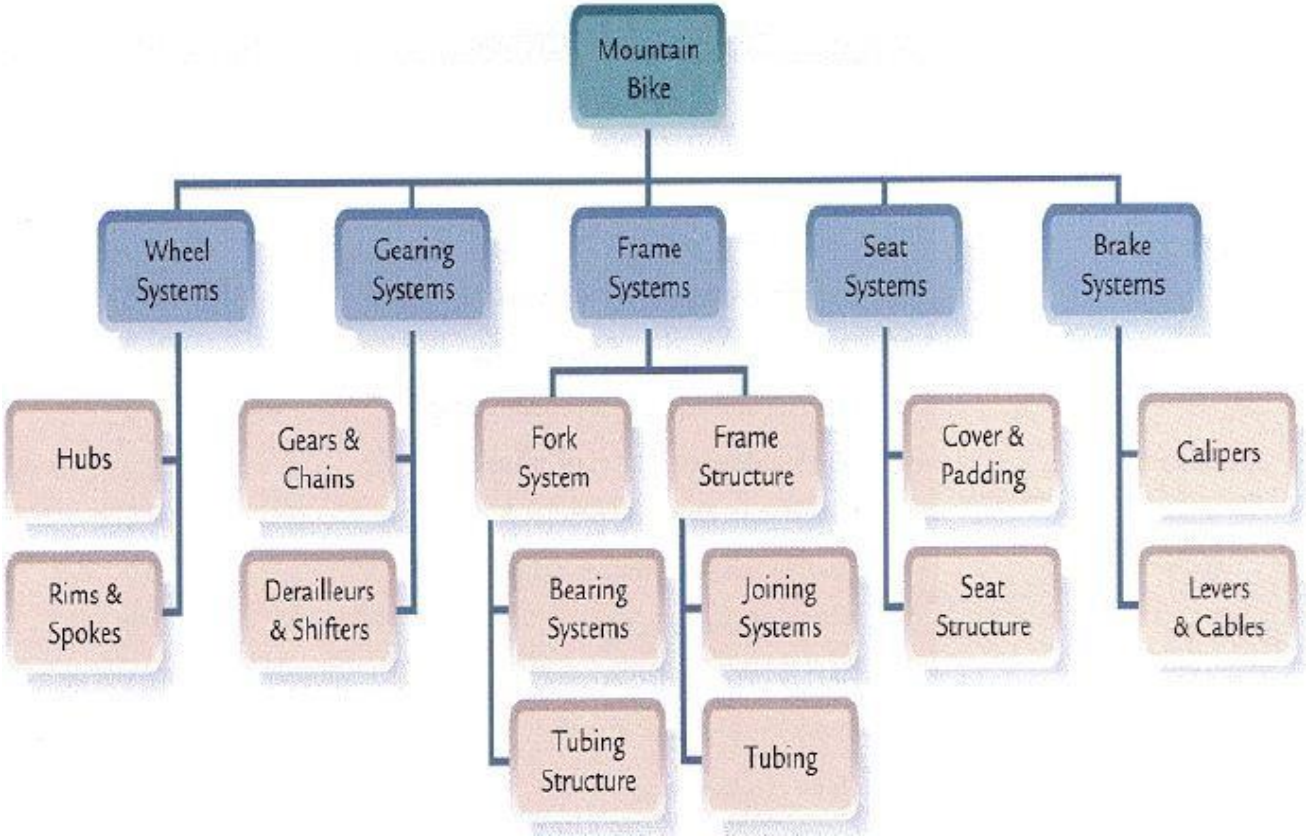
Define a baseline for performance measurement and control.

Facilitate clear responsibility assignments.

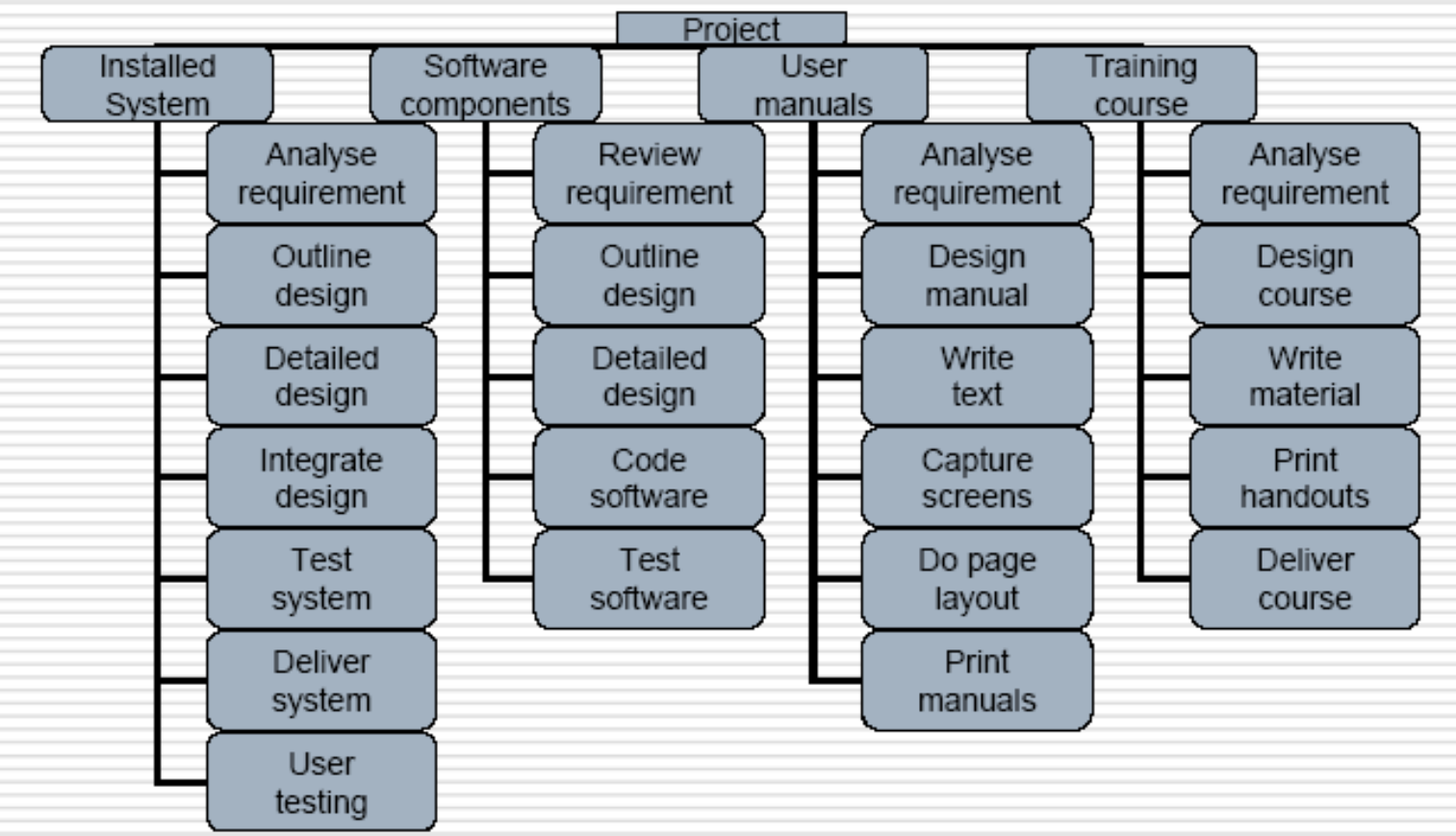
Hierarchy of WBS



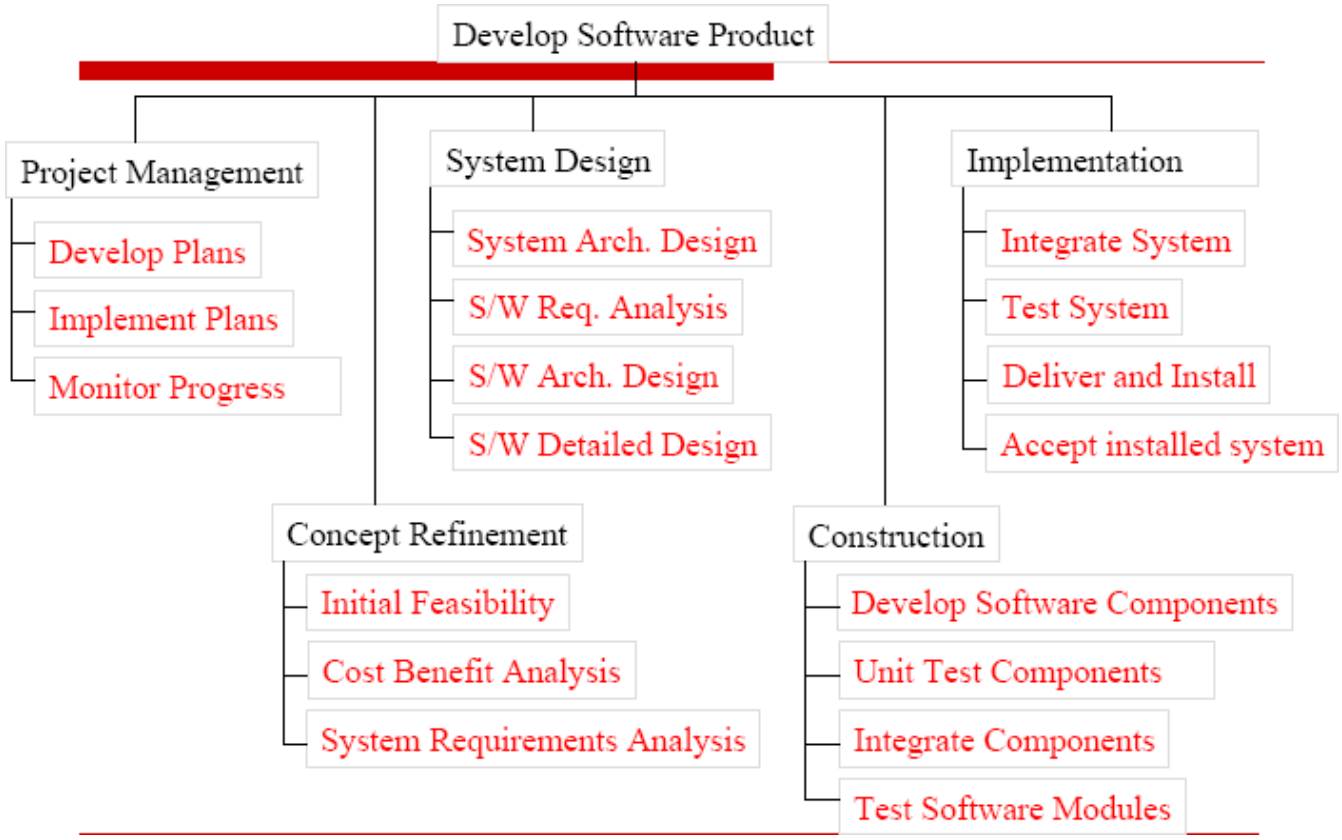
Product Based WBS



Deliverables Based WBS



WBS Template for Software Development Project

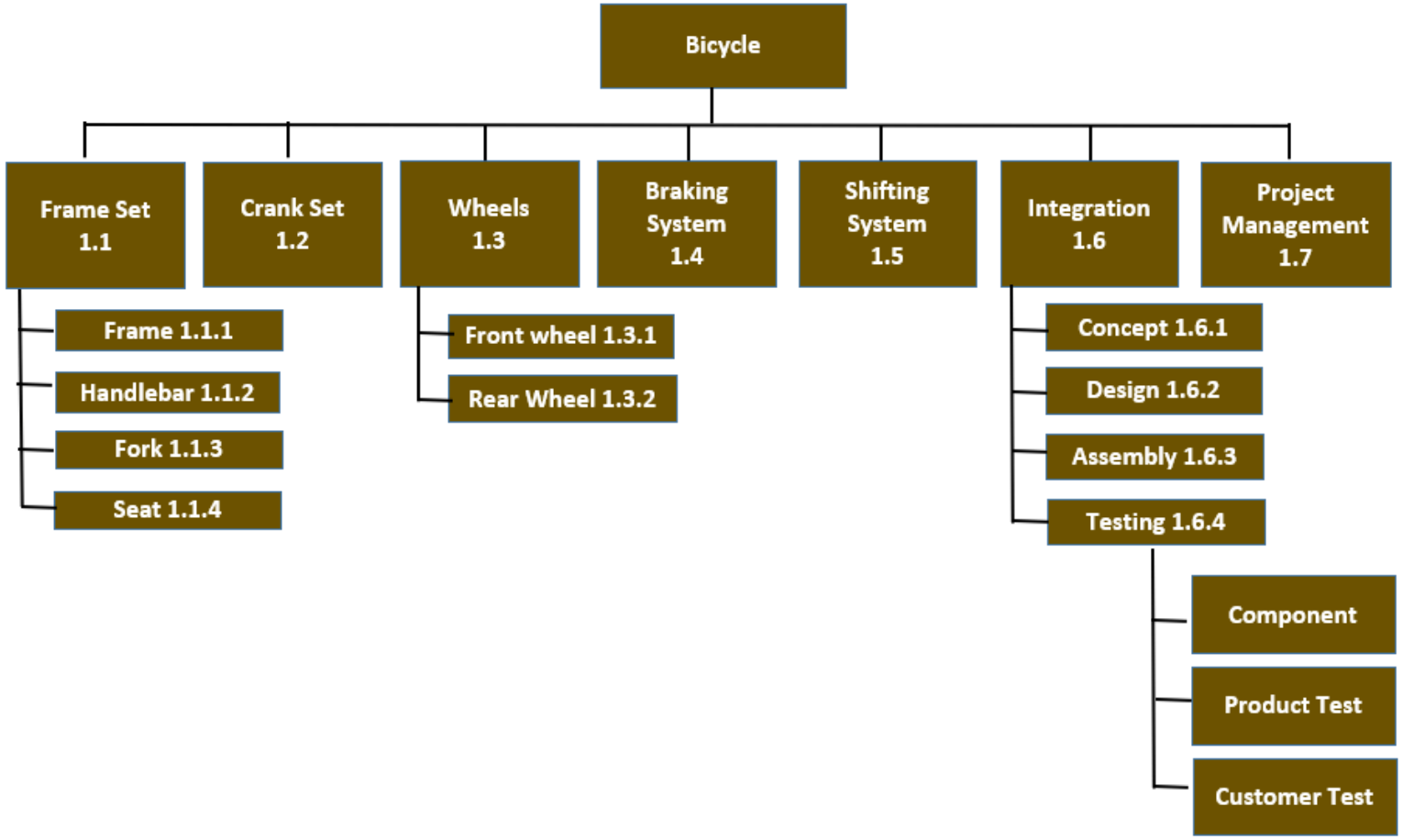


WBS Framework

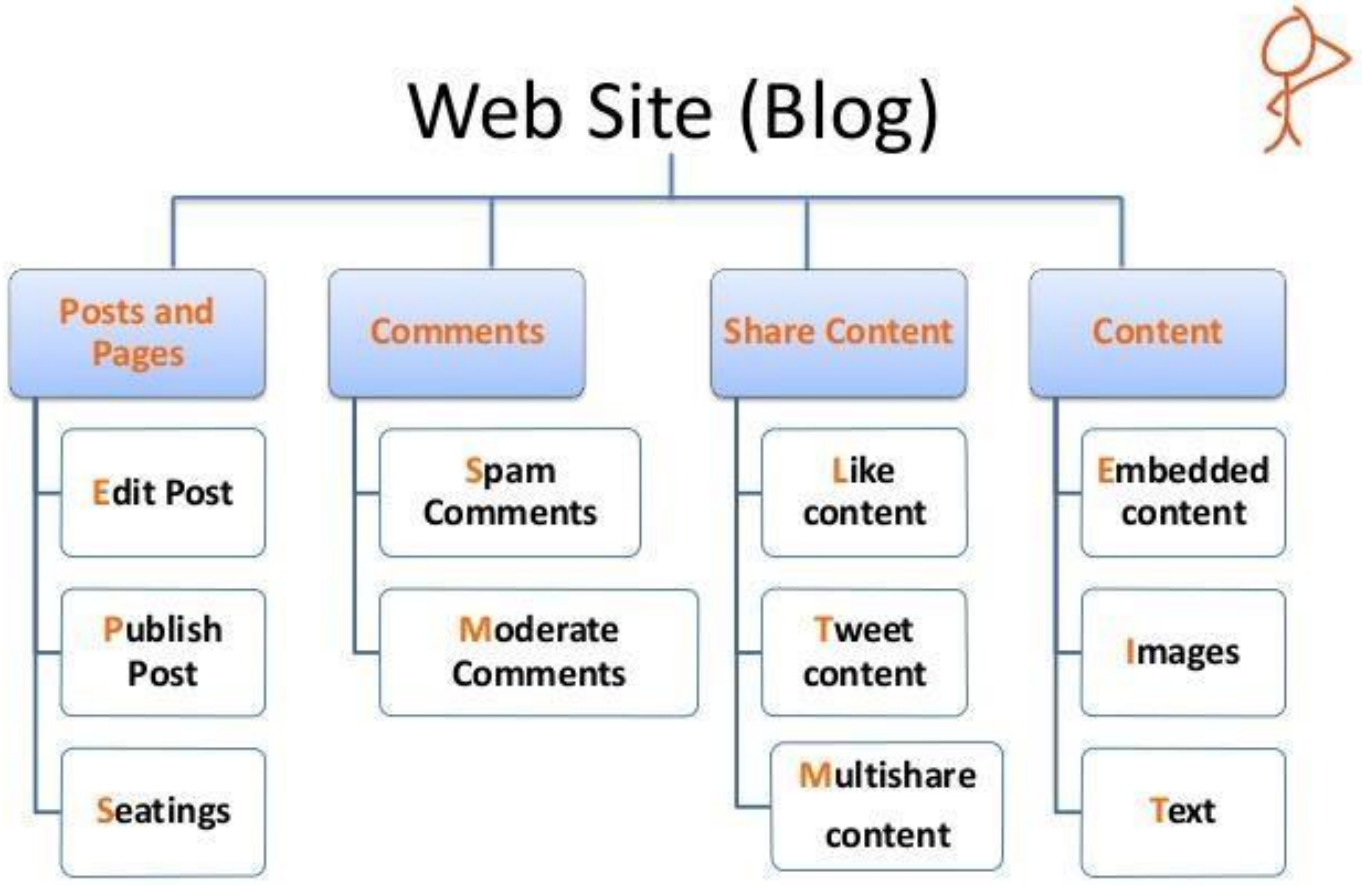
A framework dictating the number of levels and the nature of each level may be imposed on a WBS (e.g. IBM recommended **five levels):**

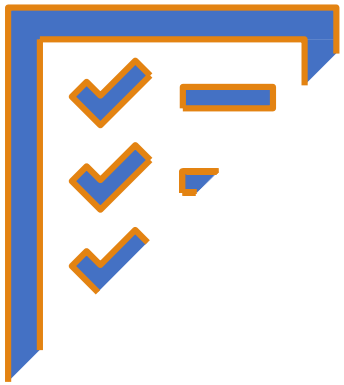
- **Level 1:** Project.
- **Level 2:** Deliverables (such as software, manuals and training courses)
- **Level 3:** Components; Which are the key work items needed to produce deliverables (e.g. modules and tests required to produce the system software)
- **Level 4:** Activities (**Work-packages** which are major work items, or collections of related tasks, required to produce a component)
- **Level 5:** Tasks (tasks that will normally be the responsibility of a single person).

Example


Bicycle
Frame Set 1.1
Crank Set 1.2
Wheels 1.3
Braking System 1.4
Shifting System 1.5
Integration 1.6
Project Management 1.7
Frame 1.1.1
Handlebar 1.1.2
Fork 1.1.3
Seat 1.1.4
Front wheel 1.3.1
Rear Wheel 1.3.2
Concept 1.6.1
Design 1.6.2
Assembly 1.6.3
Testing 1.6.4
Component
Product Test
Customer Test

Examples


Web Site (Blog)
Posts and Pages
Edit Post
Publish Post
Seatings
Comments
Spam Comments
Moderate Comments
Share Content
Like content
Tweet content
Multishare content
Content
Embedded content
Images
Text



Make a WBS for your Project

3. Activity Planning

Activity/Task List

- Tasks (Leaves of WBS/PBS as Activity Plan) and the precedence analysis (as Activity Sequencing) results in this management deliverable.
- Example format: We can make first two columns and start working on the next two (durations estimation and precedence requirements in parallel).

Activity		Duration (weeks)	Precedents
A	Hardware selection	6	
B	Software design	4	
C	Install hardware	3	A
D	Code & test software	4	B
E	File take-on	3	B
F	Write user manuals	10	
G	User training	3	E,F
H	Install & test system	2	C,D

Activity Sequencing

Activity Sequencing is the process of identifying and documenting the logical relationships between the project activities. This involves determining the order in which activities need to be performed. The main output of this process is a project schedule network diagram, which visually represents the flow of work in the project.

1. Identify Dependencies:

- **We look at the "Precedents" column in our Activity List to understand the relationships:**
  - C depends on A (A must finish before C can start)
  - D depends on B (B must finish before D can start)
  - E depends on B (B must finish before E can start)
  - G depends on E and F (Both E and F must finish before G can start)
  - H depends on C and D (Both C and D must finish before H can start)

Note: Activities A and B have no predecessors, meaning they can start concurrently (if resources allow).

2. Determine the Type of Dependency

1. Finish-to-Start (FS): Successor starts only after the predecessor finishes.
•Scenario: This is the most common type. Think of building a wall (predecessor) and then painting it (successor). You can't really start painting until the wall is built.

2. Finish-to-Finish (FF): Successor finishes only after the predecessor finishes.
•Scenario: These activities might need to conclude together, even if they start at different times. Consider writing a report (predecessor) and having it edited (successor). The final version of the report can only be released after both the writing and editing are complete. The editing might start after some of the writing is done, but it can't finish before the writing is finished.

3. Start-to-Start (SS): Successor starts only after the predecessor starts.
•Scenario: These activities can start concurrently, but the start of one is dependent on the other having begun. You're making a quick breakfast. You want to have toast ready around the same time your coffee is brewed. You don't need the coffee to be fully finished before you start the toast, but you do need to have started the coffee brewing process.

4. Start-to-Finish (SF): Successor finishes only after the predecessor starts.
•Scenario: This is the least common type and can be a bit counterintuitive. It implies that the finish of the successor activity is dependent on the start of the predecessor. Think of a security guard's shift (successor) ending only after the next guard's shift (predecessor) has started. The end of one shift is tied to the beginning of the next.

Network Planning Models

- Approaches to **scheduling;** that achieve separation between the **logical** (relationships) and the **physical** (constraints/execution); use **networks** to **model** the project.
- i.e. represent project's activities and their relationships as a network.
- first stage in creating a network model: represent the activities and their interrelationships as a graph.

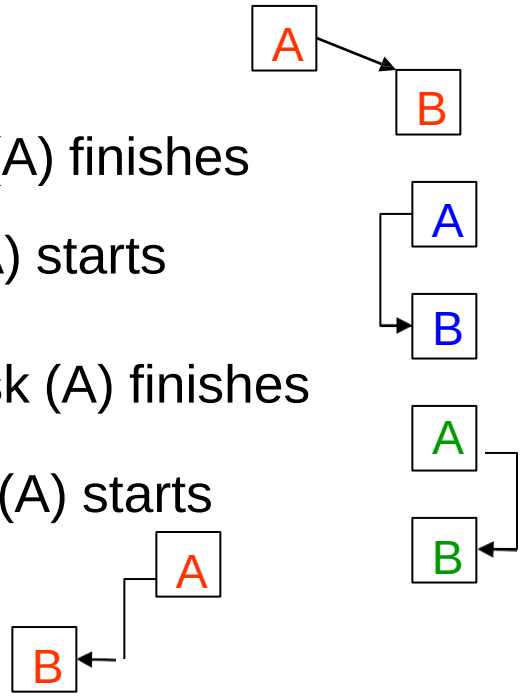
Task Dependencies (relationships)

Finish-to-start (FS): Task (B) cannot start until task (A) finishes

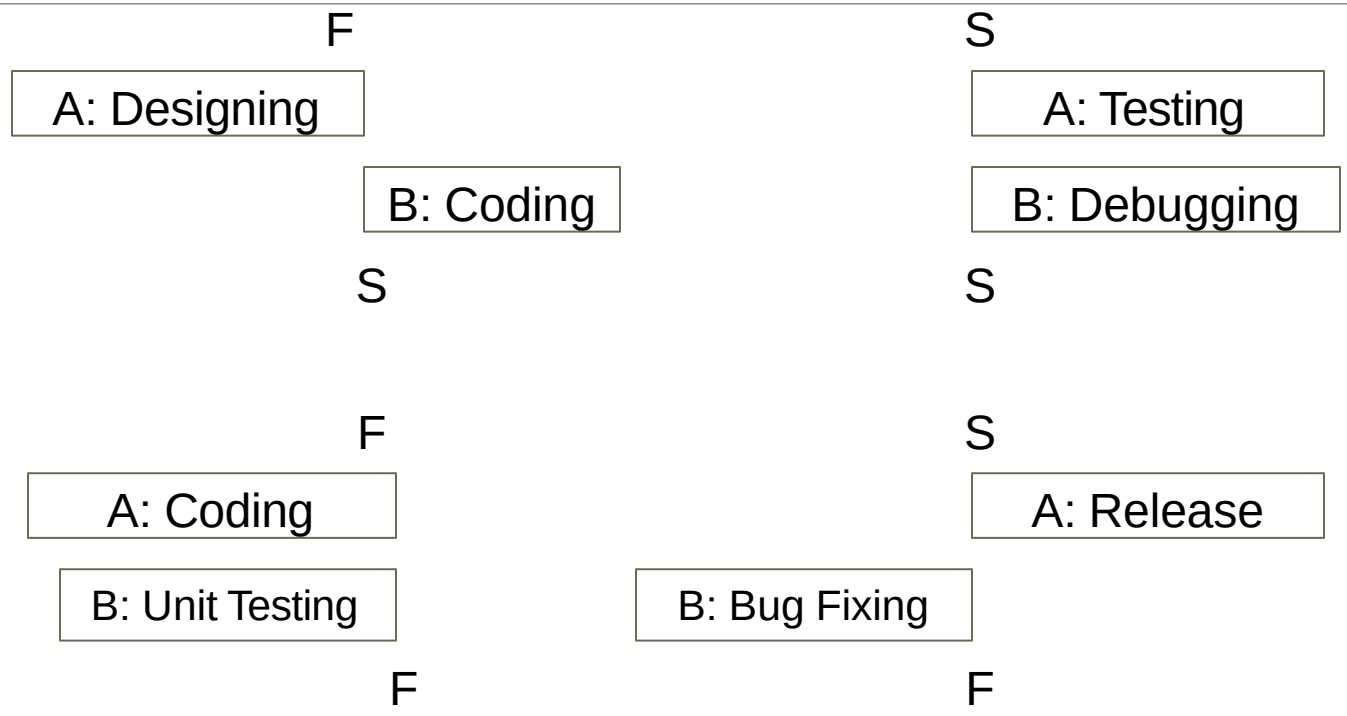
Start-to-start (SS): Task (B) cannot start until task (A) starts

Finish-to-Finish (FF): Task (B) cannot finish until task (A) finishes

Start-to-finish (SF): Task (B) cannot finish until task (A) starts



Scheduling as precedence



3. Project Network Diagrams

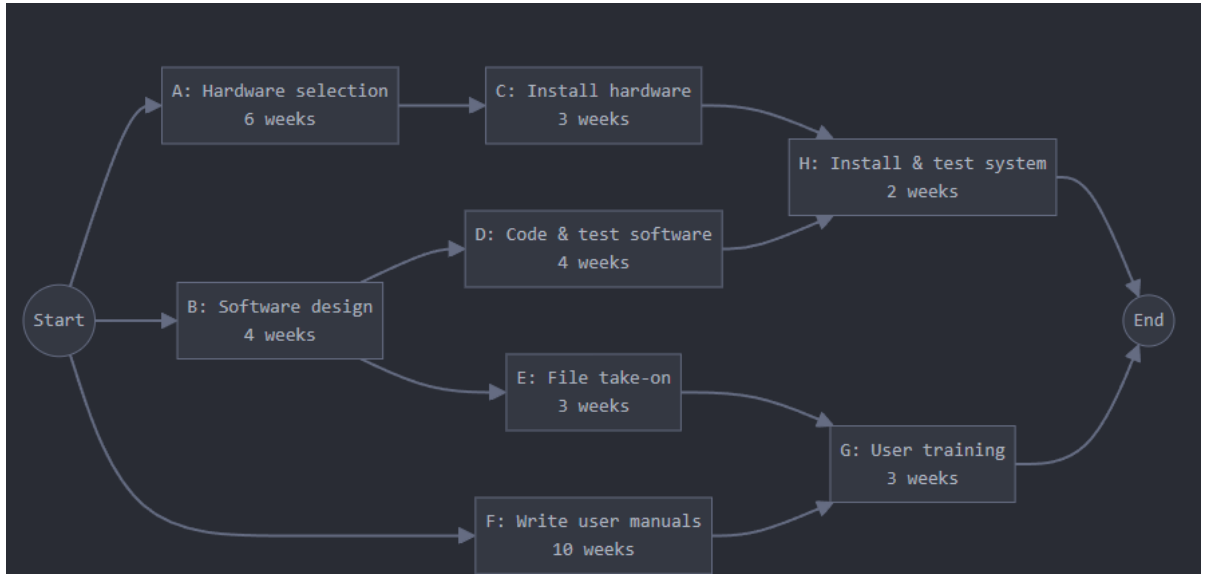
- Project network diagrams (PND) are the preferred technique for showing activity sequencing
- A project network diagram is a schematic display of the logical relationships among, or sequencing of, project activities

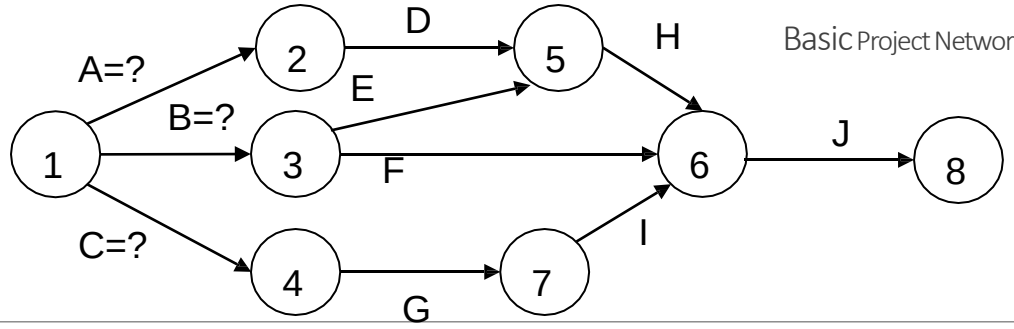
- **Activity-on-Node (AON):** Activities are represented by boxes (nodes), and dependencies are shown as arrows connecting the boxes. This is the more common method.

- **Activity-on-Arrow (AOA):** Activities are represented by arrows, and nodes represent the start and finish points of activities (milestones). This method is less commonly used now.

Activity-on-Node (AON)

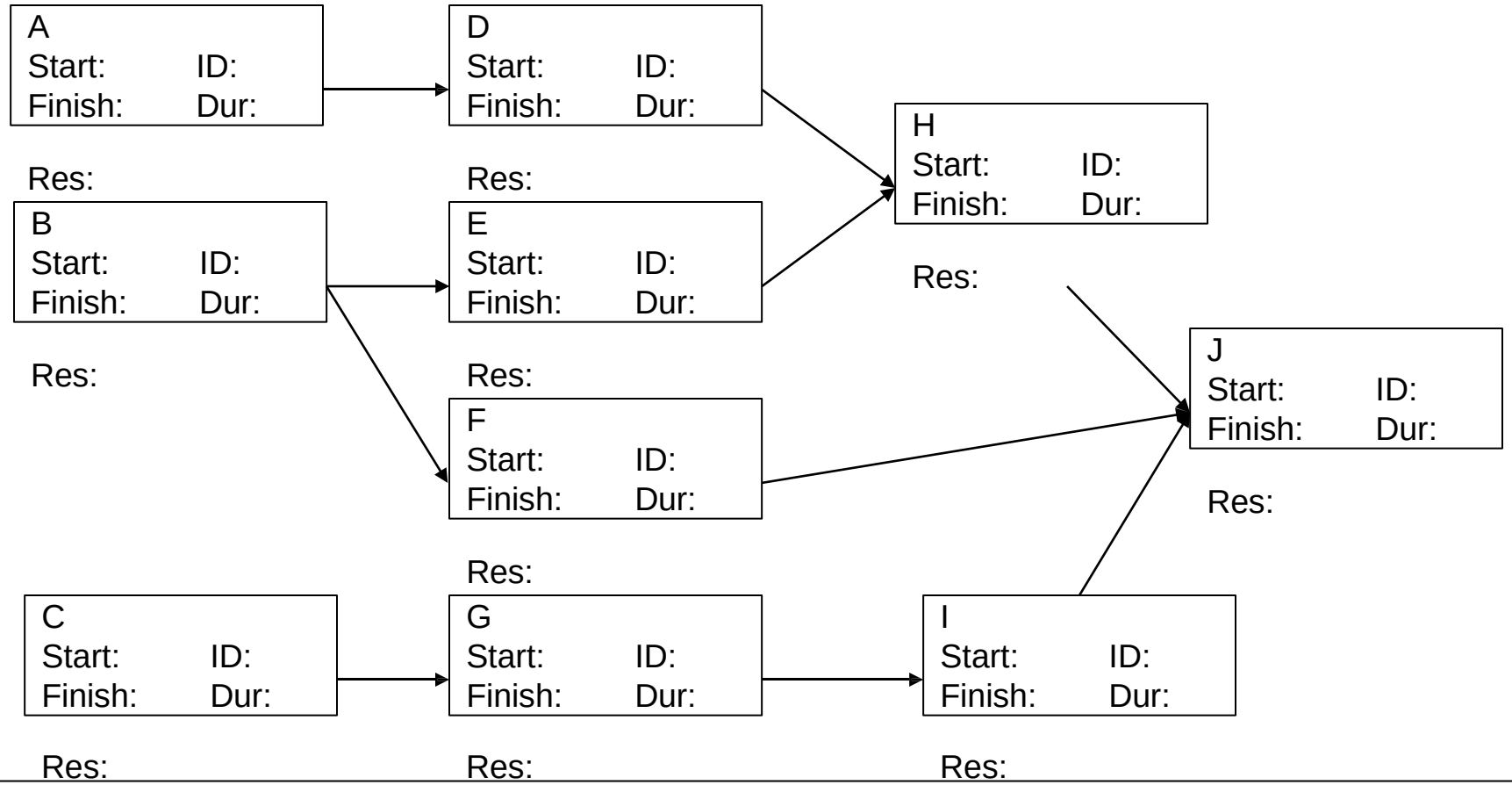
Activity		Duration (weeks)	Precedents
A	Hardware selection	6	
B	Software design	4	
C	Install hardware	3	A
D	Code & test software	4	B
E	File take-on	3	B
F	Write user manuals	10	
G	User training	3	E,F
H	Install & test system	2	C,D





Basic Project Network Diagrams (PND).....

AOA (ADM);
CPM adds more inf at nodes



Network model

- represents activities as links (arrowed lines) in the graph
- nodes (circles) represent the events of activities start and finish.

Rules for CPM network construction

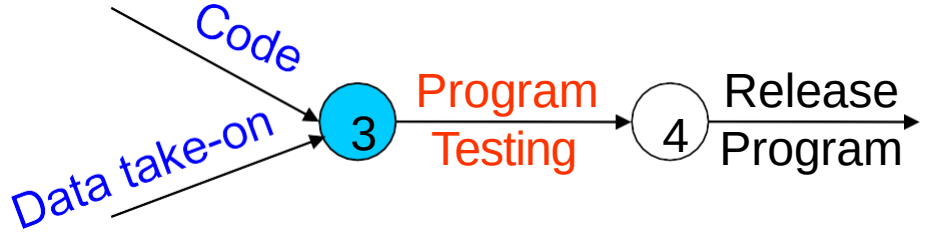
Nodes:

- A project network may have only one start node (node 1) that designates the points at which the project may start. All activities coming from that node may start immediately; resources are available.
- Network may have only one end node; designates the completion of the project and a project may only finish once.
- Nodes are events have no duration (instantaneous points in time).
- **source node:** event of the project becoming ready to start and
- **Sink node:** is the event of the project becoming completed.
- **Intermediate nodes**: represent two simultaneous events – the event of all activities (**leading in** to a node) having been **completed** and the event of all activities (**leading out** of that node) being in a position to be **started.**

A link represents an activity (and has duration)

Examples:

Node 3 is an event indicates that both “Code’ and “Data take-on” have been completed and “program Testing” can be started



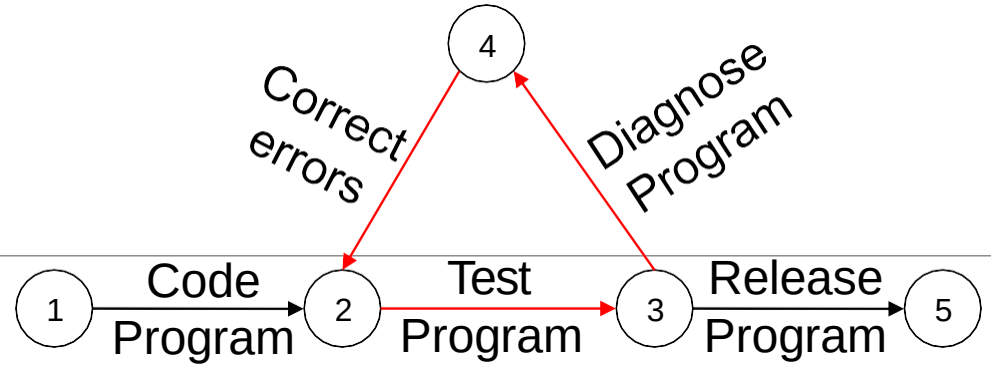
Precedents are the immediate preceding activities

both activities “Code’ and “Data take-on” are called **Precedents** of “program Testing” (**not of** ”Release Program”) and;
“program Testing” is **Precedent of** ”Release Program”.

- **Time moves from left to right**
- **Nodes are numbered sequentially**

❖ **Network may not contain loops**

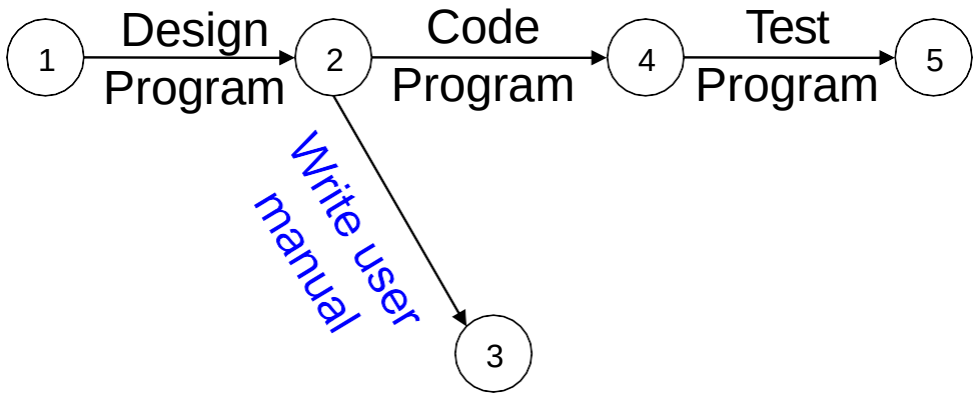
1) If we know the # of times to repeat a set of activities (e.g. test-diagnose-correct) then we can draw that set a straight sequence, repeating it the appropriate number of times.
2) If we do not know then we cannot calculate the duration of the project.



A loop represents impossible sequence

❖ **Network may not contain dangles**

A dangling activity such as "Write user manual" cannot exist, as it would suggest two completion points.



Using dummy activities

Two paths within a network have a common event although they are, in other respects independent, a logical error like the following might occur.

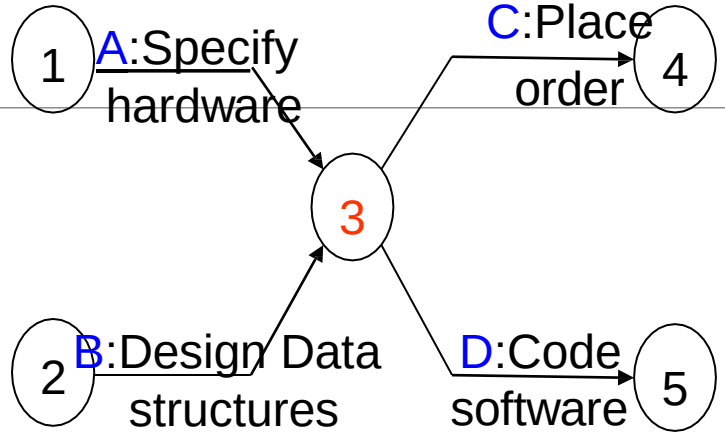
Practical Situation (Case1):

A > C (A precedent of C)

A, B > D (A & B precedent of D)

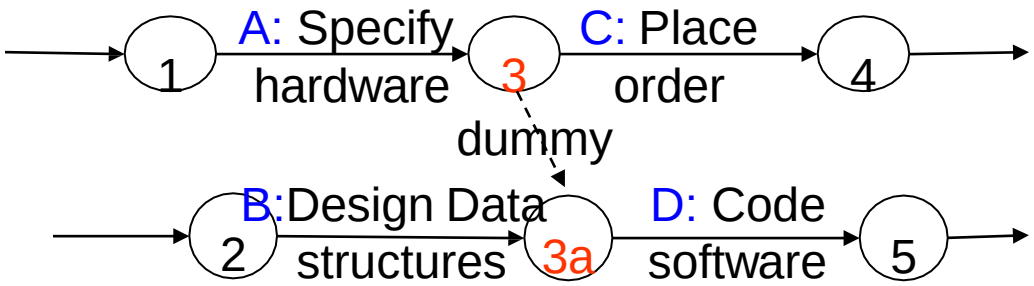
Network shows (incorrectly):

A, B > C, D (Both precedent of Both)



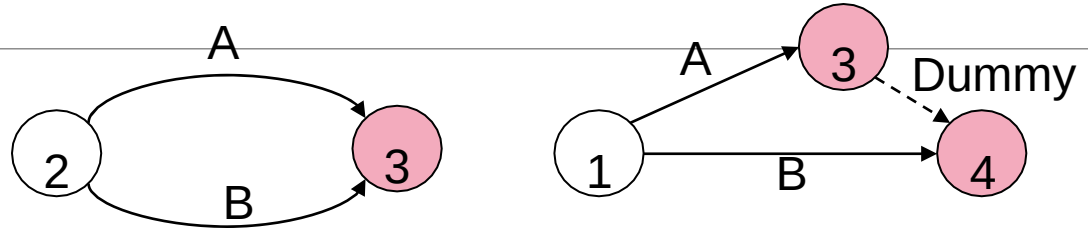
Two paths with a common node.

We can resolve this problem by separating the two (more or less) independent paths and introduce a **dummy activity to link broken event** (3). This effectively breaks unwanted link between "design data structure" and "place order".



Two paths linked by a dummy activity

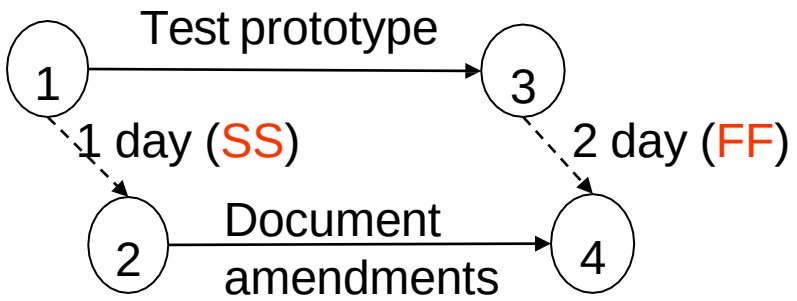
- **Dummy activities**, shown as dotted lines in the network diagram, have a zero duration and use no resources.
- **Case 2:** The **use of a dummy** activity where two activities share the same start and end nodes makes it easier to **distinguish** the activity end-points*



Representing lagged activities

- We might come across situations where we wished to undertake two activities in parallel but there is a lag between the two (time difference between start or finish).
- Impossible to show (like “amendment recording” can start after “testing” and finish a little after the completion of “testing”).
- It is better to show Each stage as a separate node.

Such parallel activities with a time lag between them are represented with pairs of dummy activities

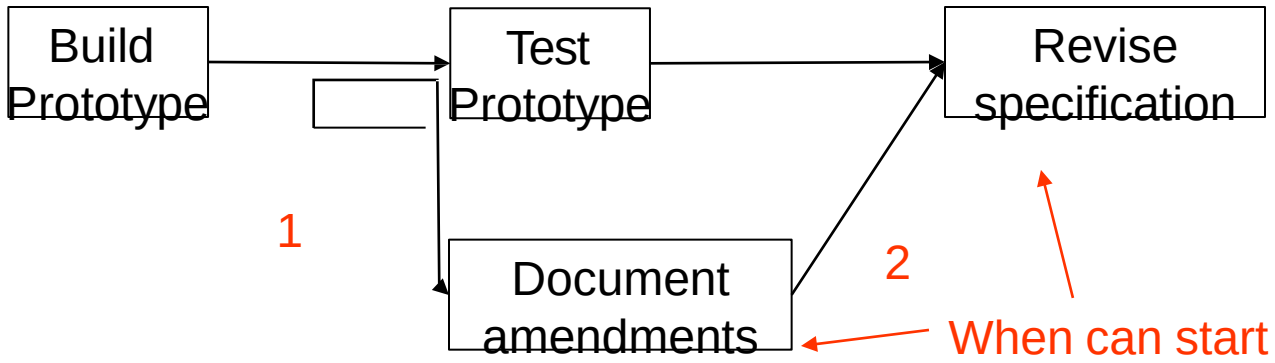


Precedence Networks (PN/PDM)

- Where CPM networks use links to represent activities and nodes to represent events, **precedence networks** use **boxes** (nodes) to represent **activities** (known as work items) and **links** to represent **dependencies**.
- The **boxes** may carry task descriptions and duration estimates and the **links** may contain a duration denoting a lag between the completion/start of the next.
- It contains much more information than the CPM network and we do not need to keep a separate activity table.
- Analysis of precedence networks proceeds in exactly the same ways.

An other advantage of PN is that they can represent parallel lagged activities (Which required use of dummy activities in CPM network) much more elegantly.

Parallel lagged activities in a precedence network.



Schedule development :Adding the time dimension

Moving from Logical to Physical network model:

- we are now ready to start thinking about when each activity should be undertaken (Physically).

CPM

project network analysis technique used to predict total project Duration and concerned with two primary objectives:

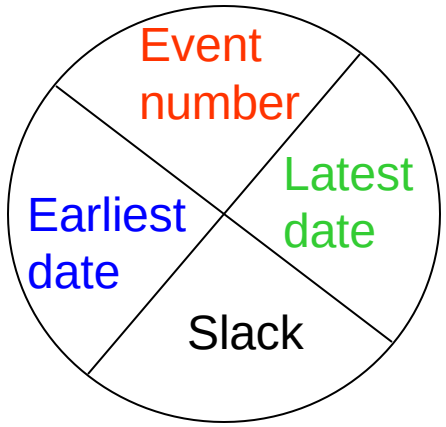
- Planning the project in such a way that it is completed as quickly as possible; and
- Identifying those activities where a delay in their execution is likely to affect the overall end date of the project or 'later activities' start dates.

Network Analysis: The network is then analyzed by carrying out:

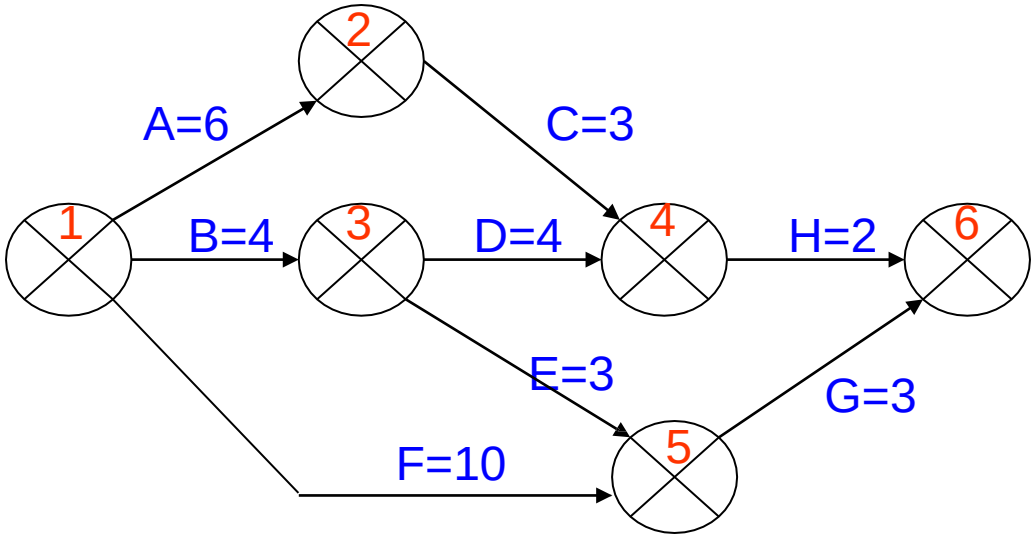
- **forward pass**, to calculate the earliest dates at which activities may commence and the project be completed, and a
- **backward pass**, to calculate the latest start dates for activities and the critical path.

Constructing CPM Network

- Typically information about **events** is recorded on the network (and **activity-based** information is generally held on a separate activity table).
- common convention is to divide the node circle into quadrants to show the event number, the latest and earliest dates, and the event slack.



CPM node

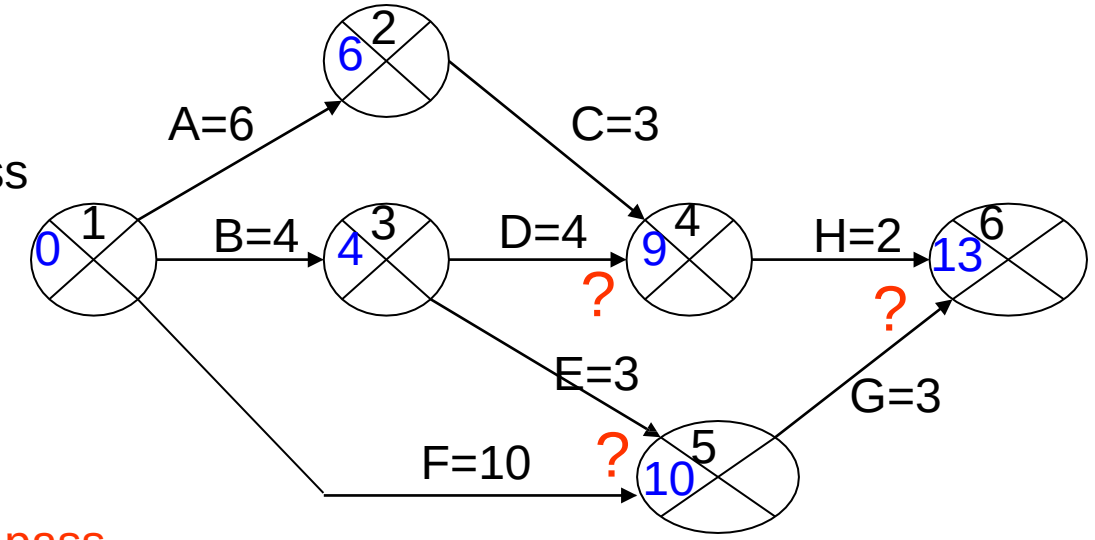


The forward pass

- carried out to calculate the earliest date on which **each event** may be achieved and the earliest dates on which **each activity** may be started and completed.
- **Earliest** dates for **events** are recorded on the network diagram and for **activities** on the activity table.

Forward Pass

CPM network after forward pass



Activity table after the forward pass

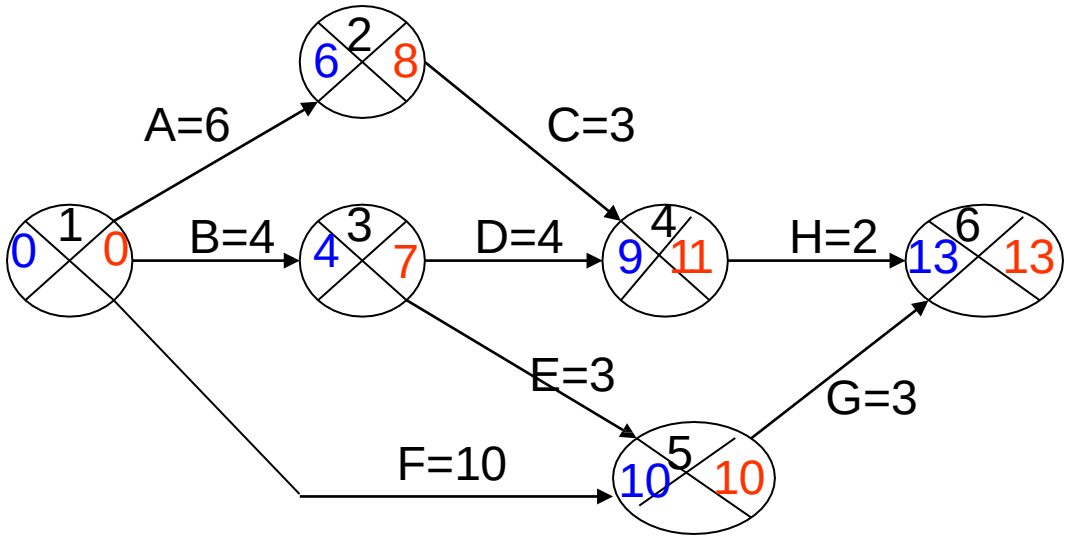
Activity	Duration (weeks)	Earliest start date	Latest start date	Earliest finish date	Latest finish date	
A	6	0		6		
B	4	0		4		
C	3	6		9		
D	4	4		8		
E	3	4		7		
F	10	0		10		
G	3	10		13		
H	2	9		11		

Backward Pass

- to calculate the **latest date** at which each event may be achieved, and each activity started and finished, **without delaying** the end date of the **project.**
- we assume that the latest finish date for the project is the same as the earliest finish date.

Rule:

- The latest date for an event is the latest start date for all activities commencing from that event.
- In case more activities, we take the earliest of the latest start dates for those activities. (e.g. latest start dates for A=2, B=3, F=0; and earliest among all =0 for event#1)



CPM network after backward pass

Up date the activity table

For Latest start and finish dates

Activity	Duration (weeks)	Earliest start date	Latest start date	Earliest finish date	Latest finish date	
A	6	0	2	6	8	
B	4	0	3	4	7	
C	3	6	8	9	11	
D	4	4	7	8	11	
E	3	4	7	7	10	
F	10	0	0	10	10	
G	3	10	10	13	13	
H	2	9	11	11	13	

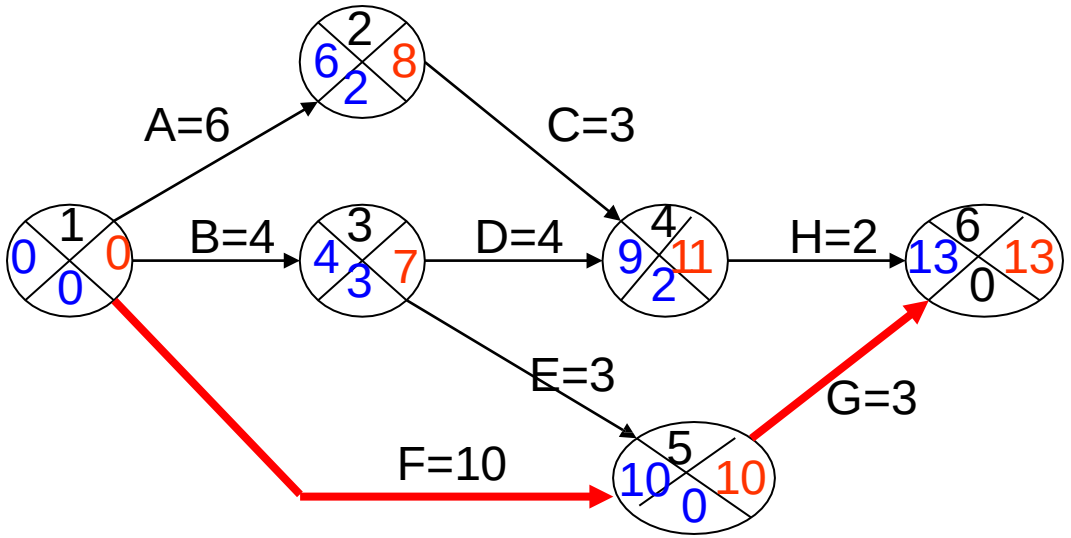
Critical Path

- **A critical path for a project is the series of activities that determines the *earliest time* by which the project can be completed.**
- **The critical path is the *longest path* through the network diagram and has the least amount of slack or float.**

- Any delay on the critical path will delay the project.
- **Slack:** The difference between the earliest date and the latest date for an event – measure of how late an event may be **without affecting** the **end date** of the project.
- Any event with a **slack of zero** is **critical:** any delay in achieving that event will delay the completion date of the project as a whole.
- There will always be at least one path through the network joining those **critical events** – this path is known as the **critical path.**

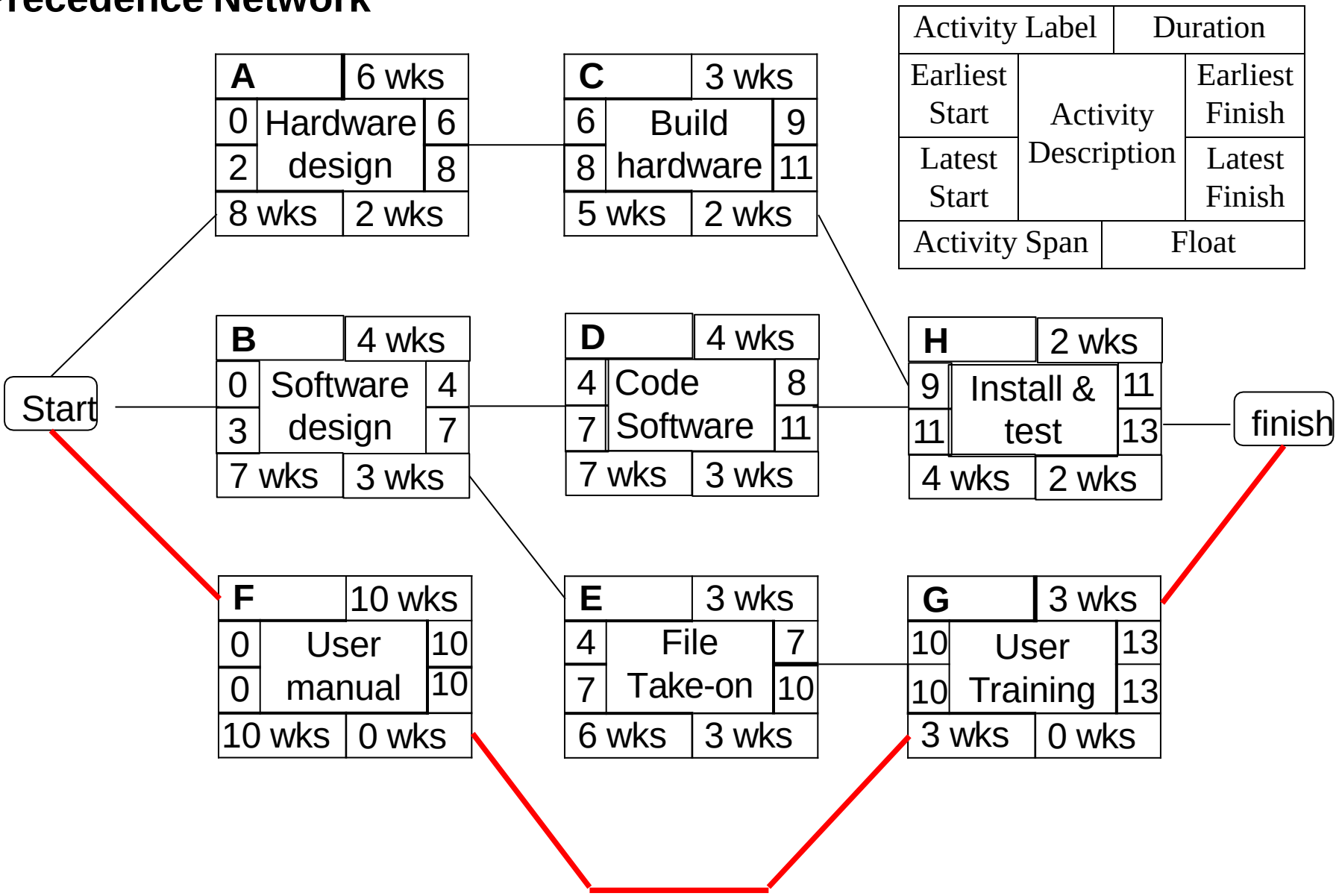
significance of critical path is two-fold.

1. **In planning:** it is the critical path that we must shorten if we are to reduce the overall duration
2. **In managing**: must pay attention to monitoring activities on the critical path so that the effects of any delay or resource unavailability are detected at the earliest opportunity.



The Critical path

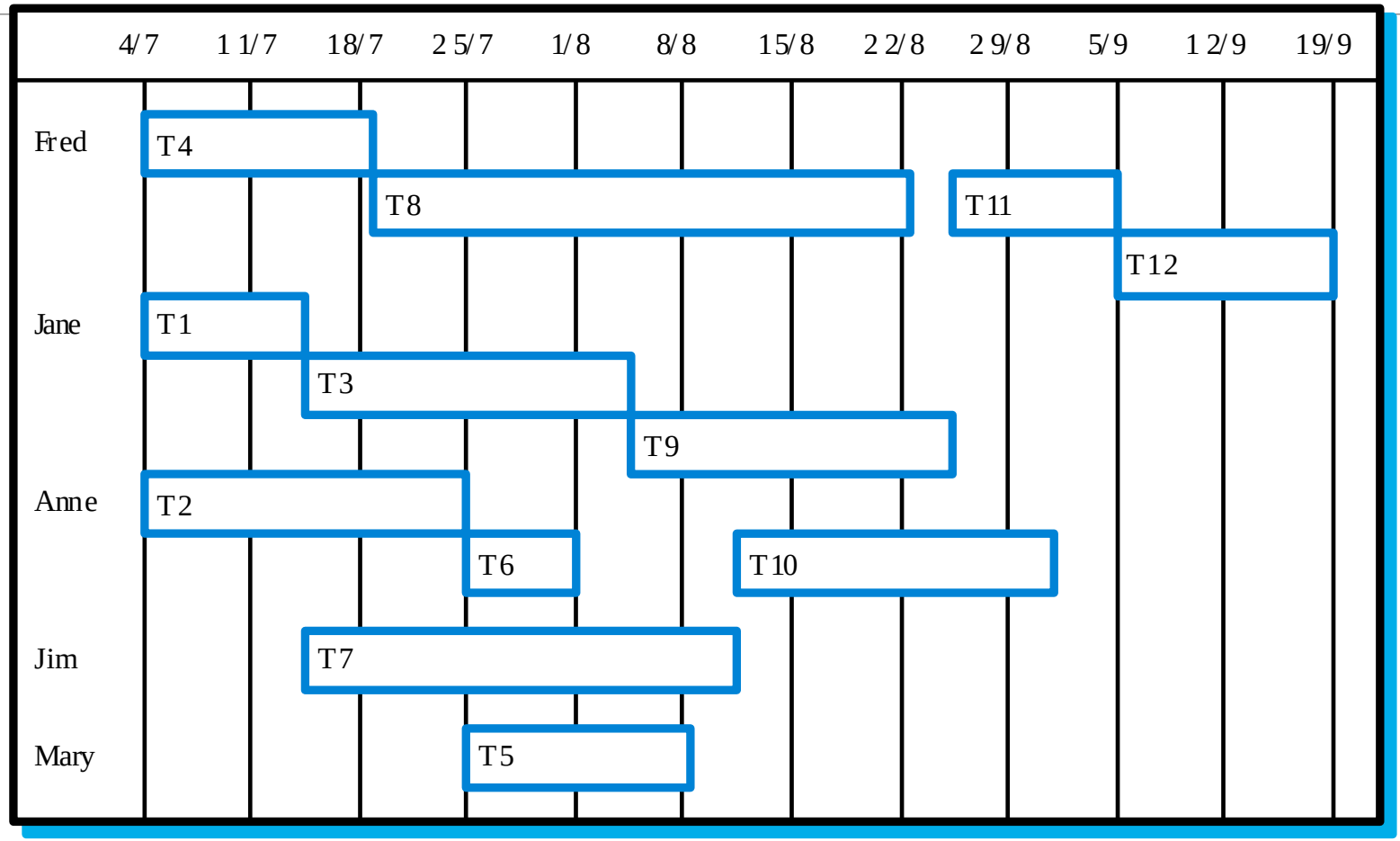
Precedence Network



The critical path through activities F and G is shown as a heavy line.

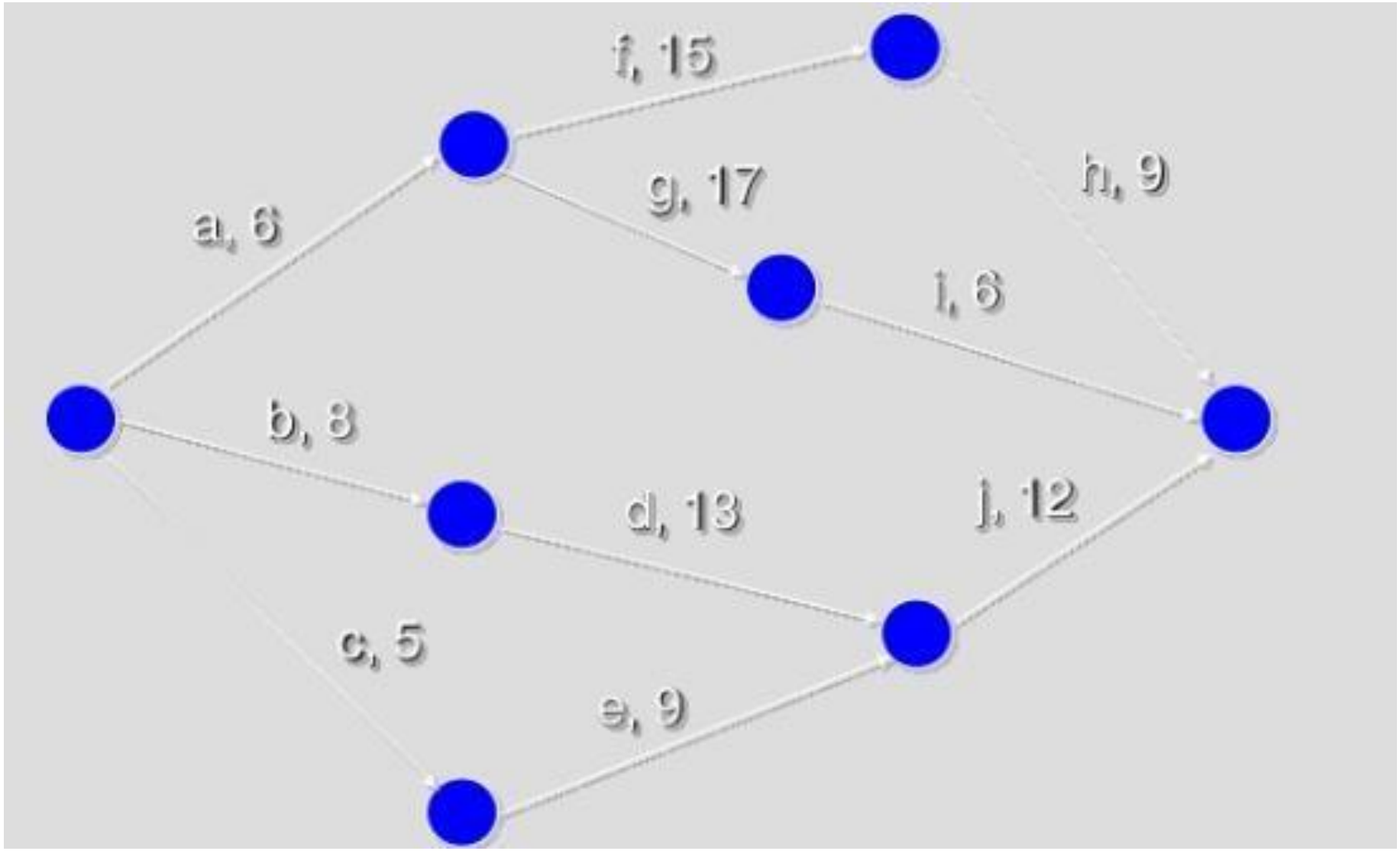
Gantt Chart

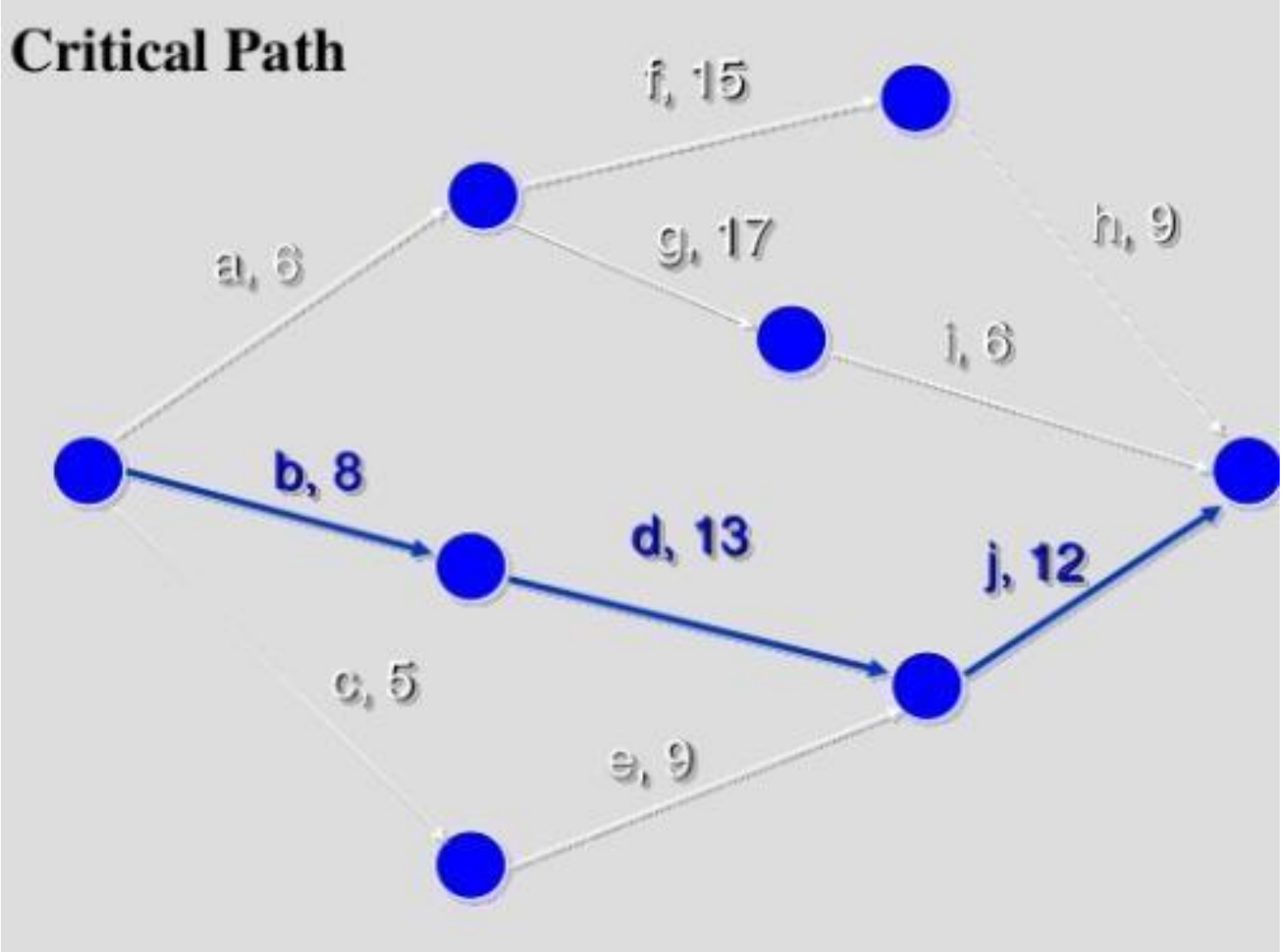
Activity Schedule and Staff allocation



Activity	Duration	Precedence
A	6	
B	8	
C	5	
D	13	B
E	9	C
F	15	A
G	17	A
H	9	F
I	6	G
J	12	D,E

Example


f, 15
g, 17
h, 9
a, 6
i, 6
b, 8
d, 13
j, 12
c, 5
e, 9


Critical Path
f, 15
a, 6
g, 17
h, 9
i, 6
b, 8
d, 13
j, 12
c, 5
e, 9


Thank
You

Activity		Duration (weeks)	Precedents
A	Hardware selection	6	
B	Software design	4	
C	Install hardware	3	A
D	Code & test software	4	B
E	File take-on	3	B
F	Write user manuals	10	
G	User training	3	E
H	Install & test system	2	C