Unit IV

PROJECT SCHEDULING AND CONTROL:

MANAGEMENT SPECTRUM, 3PS, PROCESS AND PROJECT METRICS

SCHEDULING:

WORK BREAKDOWN STRUCTURE, NETWORK DIAGRAM, GANTT CHART

Chapter 21 Project Management: 21.1, [21.2 – for reading]
Chapter 22: Metrics for Process and projects: 22.2
Chapter 24: Project Scheduling: 24.2, 24.3, 24.4, 24.5

Kiran Bhowmick

Kiran Bhowmick

Management Spectrum

- People — the most important element of a successful project
- Product — the software to be built
- Process — the set of framework activities and software engineering tasks to get the job done
- Project — all work required to make the product a reality

Stakeholders

- *Senior managers* who define the business issues that often have significant influence on the project.
- *Project (technical) managers* who must plan, motivate, organize, and control the practitioners who do software work.
- *Practitioners* who deliver the technical skills that are necessary to engineer a product or application.
- *Customers* who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
- *End-users* who interact with the software once it is released for production use.

Team Leader

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

Software Teams

How to lead?

How to organize?

How to collaborate?

How to motivate?

How to create good ideas?

Software Teams

The following factors must be considered when selecting a software project team structure ...

- the difficulty of the problem to be solved
- the size of the resultant program(s) in lines of code or function points
- the time that the team will stay together (team lifetime)
- the degree to which the problem can be modularized
- the required quality and reliability of the system to be built
- the rigidity of the delivery date
- the degree of sociability (communication) required for the project

Organizational Paradigms

- closed paradigm—structures a team along a traditional hierarchy of authority
- random paradigm—structures a team loosely and depends on individual initiative of the team members
- open paradigm—attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm
- synchronous paradigm—relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves

Avoid Team "Toxicity"

- A frenzied work atmosphere in which team members waste energy and lose focus on the objectives of the work to be performed.
- High frustration caused by personal, business, or technological factors that cause friction among team members.
- "Fragmented or poorly coordinated procedures" or a poorly defined or improperly chosen process model that becomes a roadblock to accomplishment.
- Unclear definition of roles resulting in a lack of accountability and resultant finger-pointing.
- "Continuous and repeated exposure to failure" that leads to a loss of confidence and a lowering of morale.

Problem Decomposition

- Sometimes called *partitioning* or *problem elaboration*
- Once scope is defined …
  - It is decomposed into constituent functions
  - It is decomposed into user-visible data objects

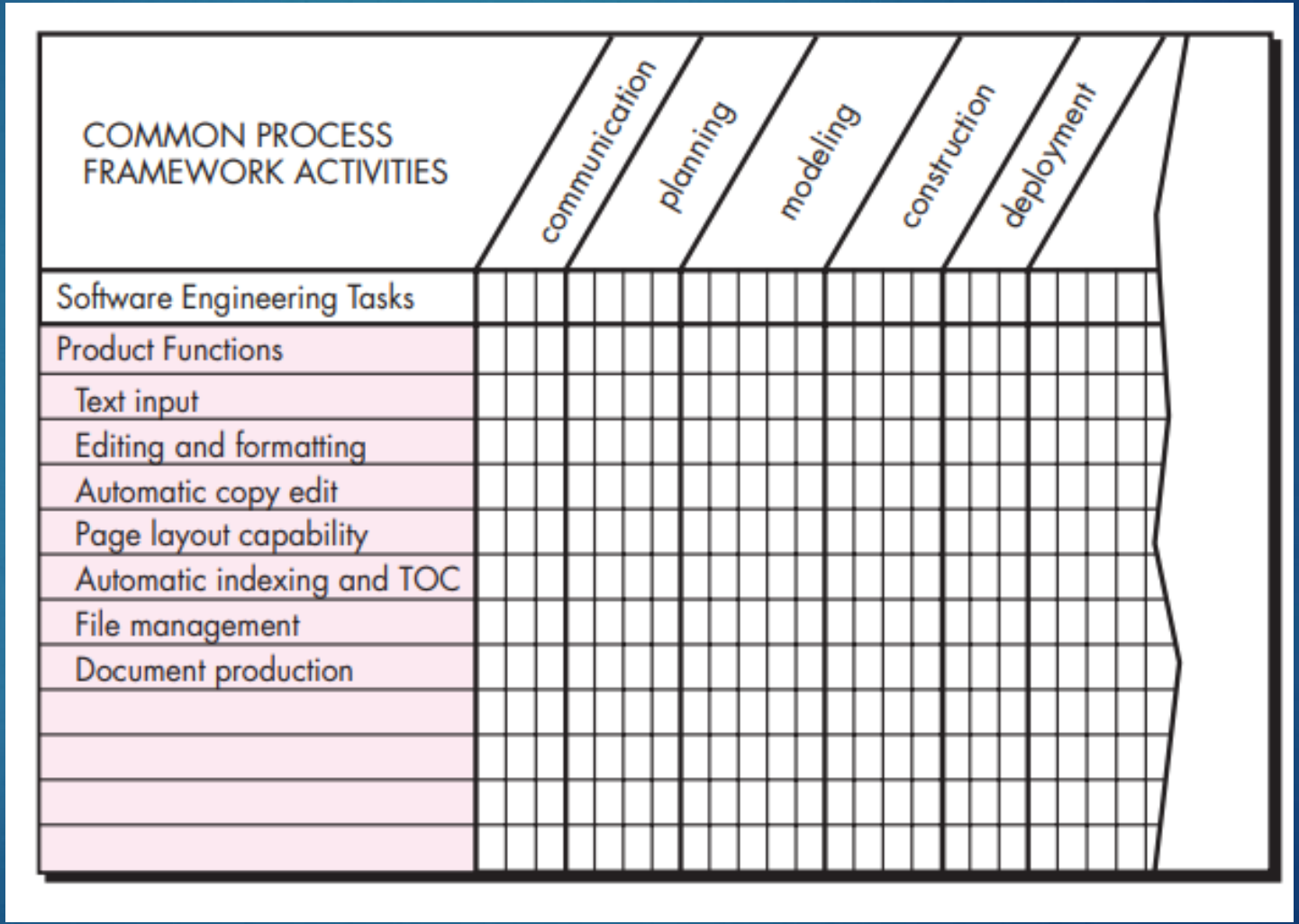
  or

  - It is decomposed into a set of problem classes
- Decomposition process continues until all functions or problem classes have been defined

The Process

- Once a process framework has been established
  - Consider project characteristics
  - Determine the degree of rigor required
  - Define a task set for each software engineering activity
    - Task set =
      - Software engineering tasks
      - Work products
      - Quality assurance points
      - Milestones

Melding the Problem and the Process



The Project

- Projects get into trouble when …
  - Software people don't understand their customer's needs.
  - The product scope is poorly defined.
  - Changes are managed poorly.
  - The chosen technology changes.
  - Business needs change [or are ill-defined].
  - Deadlines are unrealistic.
  - Users are resistant.
  - Sponsorship is lost [or was never properly obtained].
  - The project team lacks people with appropriate skills.
  - Managers [and practitioners] avoid best practices and lessons learned.

Common-Sense Approach to Projects

- *Start on the right foot.* This is accomplished by working hard (very hard) to understand the problem that is to be solved and then setting realistic objectives and expectations.
- *Maintain momentum. The* project manager must provide incentives to keep turnover of personnel to an absolute minimum, the team should emphasize quality in every task it performs, and senior management should do everything possible to stay out of the team's way.
- *Track progress.* For a software project, progress is tracked as work products (e.g., models, source code, sets of test cases) are produced and approved (using formal technical reviews) as part of a quality assurance activity.
- *Make smart decisions.* In essence, the decisions of the project manager and the software team should be to "keep it simple."
- *Conduct a postmortem analysis.* Establish a consistent mechanism for extracting lessons learned for each project.

To Get to the Essence of a Project

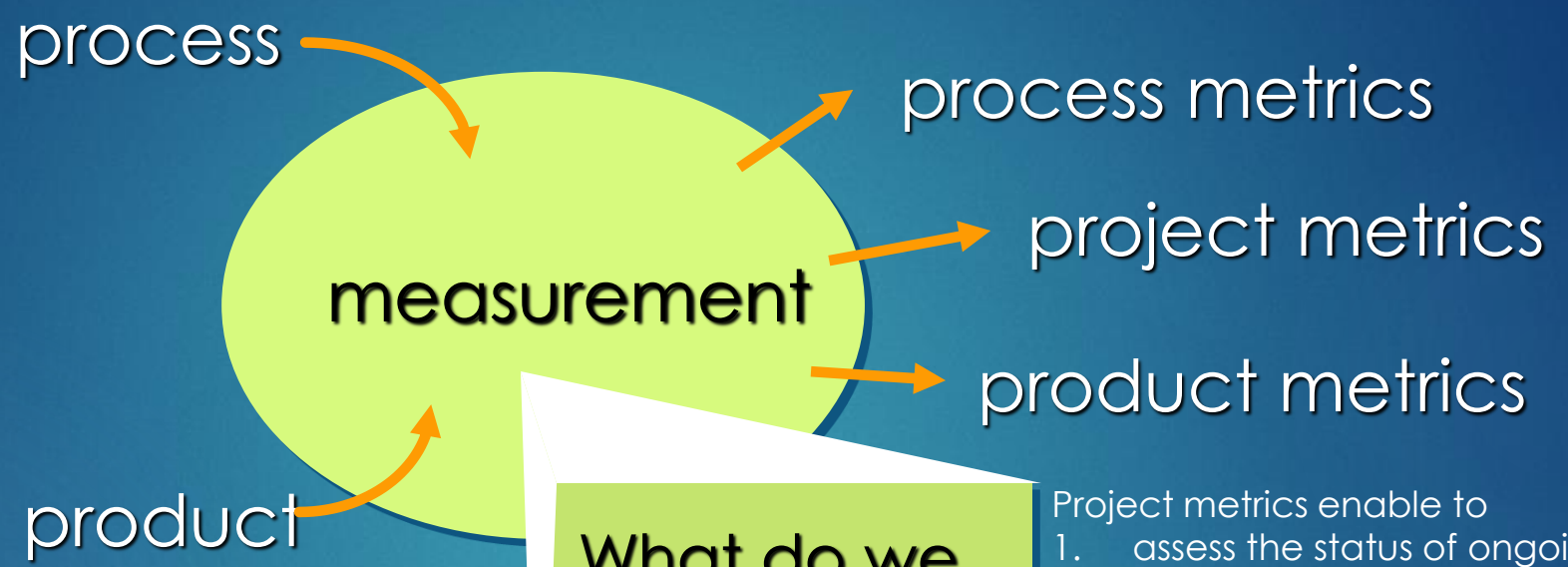
- Why is the system being developed?
- What will be done?
- When will it be accomplished?
- Who is responsible?
- Where are they organizationally located?
- How will the job be done technically and managerially?
- How much of each resource (e.g., people, software, tools, database) will be needed?

Barry Boehm

Critical Practices

- Formal risk management
- Empirical cost and schedule estimation
- Metrics-based project management
- Earned value tracking
- Defect tracking against quality targets
- People aware project management

A Good Manager Measures



Process metrics are collected across all projects and over long periods of time. Their intent is to provide a set of process indicators that lead to long-term s/w process improvement

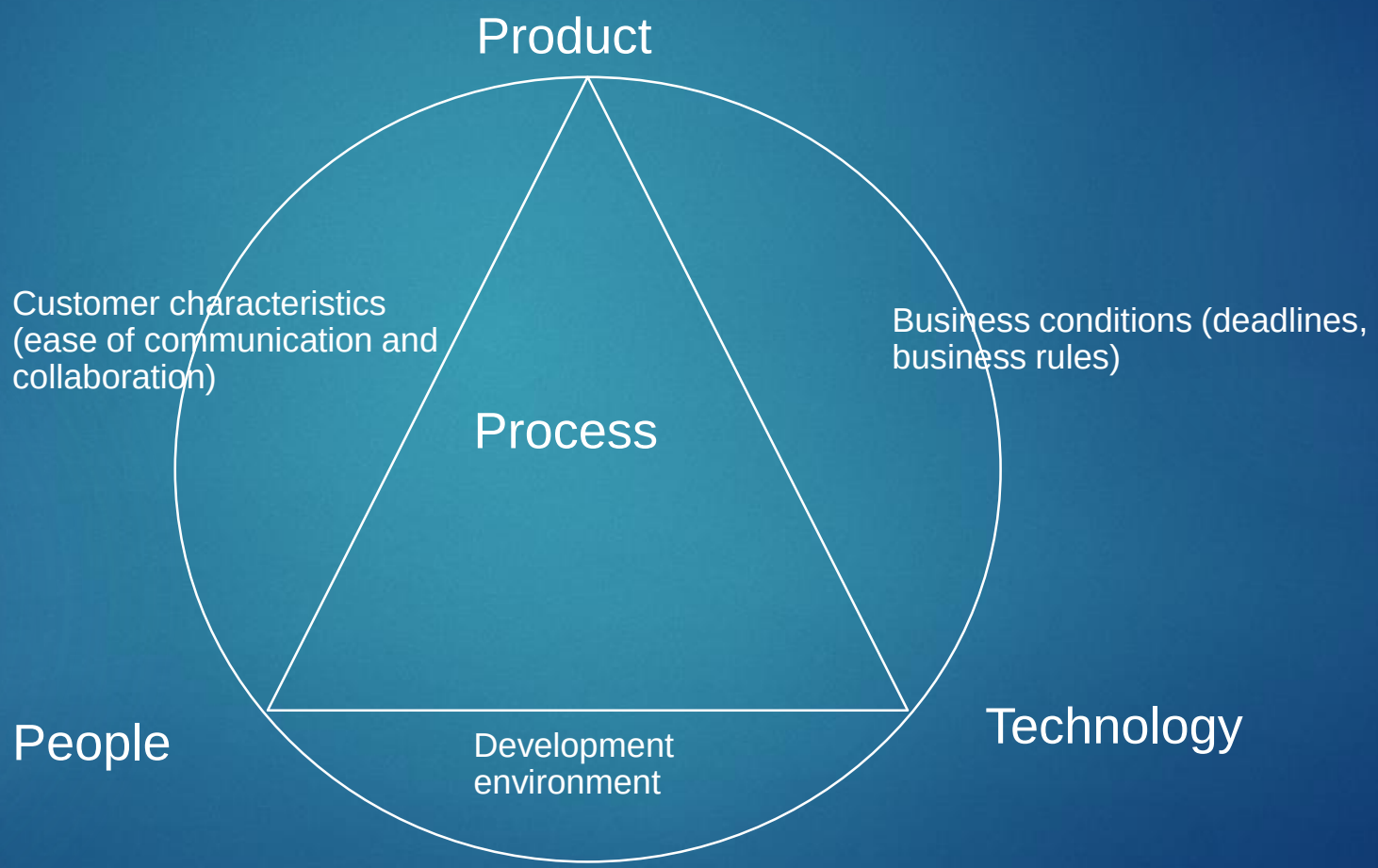
What do we use as a basis?

- size?
- function?

Project metrics enable to

1. assess the status of ongoing project
2. Track potential risks
3. Uncover problem areas before they go critical
4. Adjust work flow or tasks
5. Evaluate the project team's ability to control quality of software

Determinants of Software quality, organizational effectiveness



Process Metrics

- Quality-related
  - focus on quality of work products and deliverables
- Productivity-related
  - Production of work-products related to effort expended
- Statistical SQA data
  - error categorization & analysis
- Defect removal efficiency
  - propagation of errors from process activity to activity
- Reuse data
  - The number of components produced and their degree of reusability

Project Metrics

- used to minimize the development schedule by making the adjustments necessary to avoid delays and mitigate potential problems and risks
- used to assess product quality on an ongoing basis and, when necessary, modify the technical approach to improve quality.
- every project should measure:
  - *inputs*—measures of the resources (e.g., people, tools) required to do the work.
  - *outputs*—measures of the deliverables or work products created during the software engineering process.
  - *results*—measures that indicate the effectiveness of the deliverables.

Typical Project Metrics

- Effort/time per software engineering task
- Errors uncovered per review hour
- Scheduled vs. actual milestone dates
- Changes (number) and their characteristics
- Distribution of effort on software engineering tasks

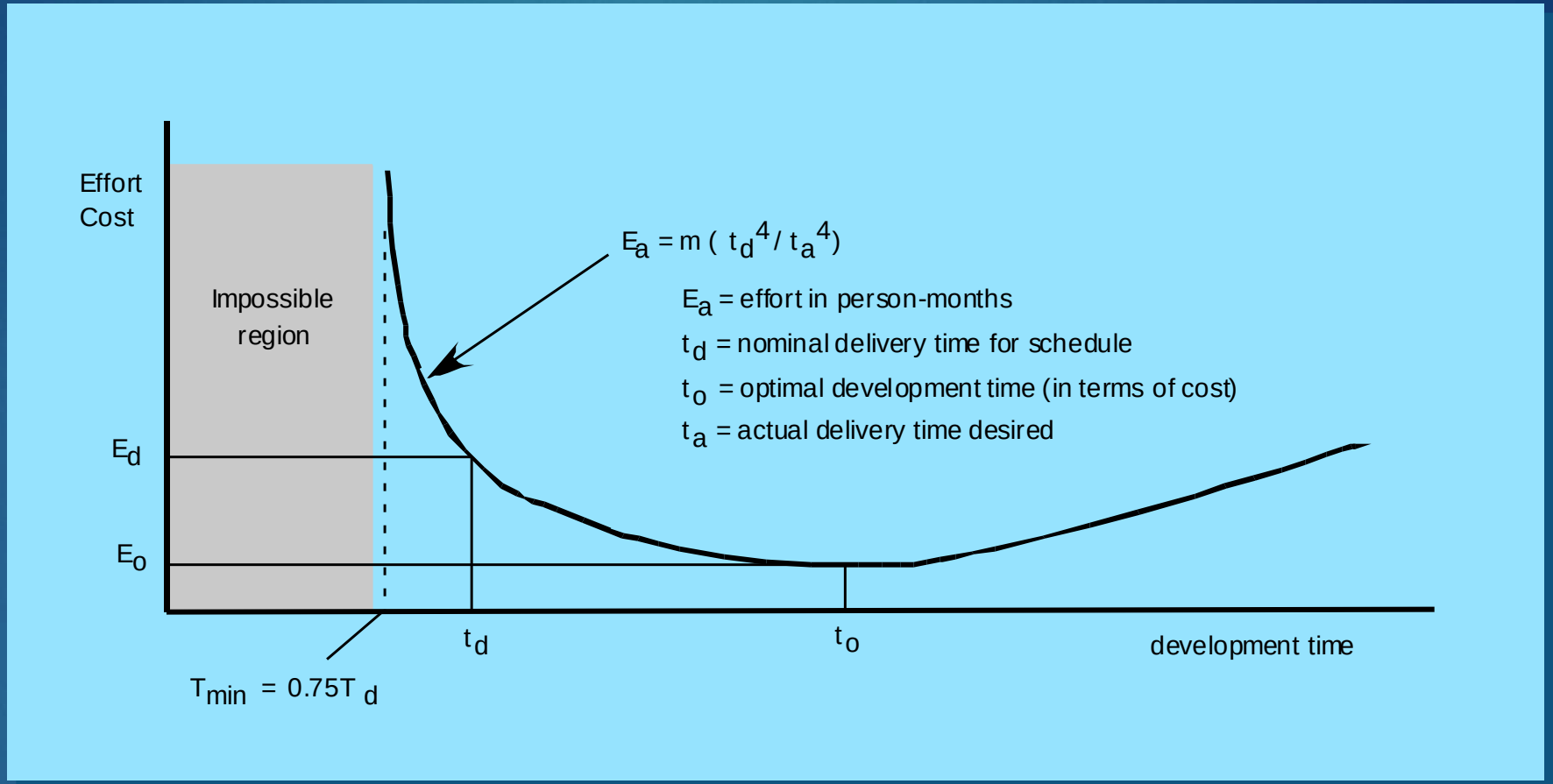
Project Scheduling

- Is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks.
- Schedules evolve over time.
- Macroscopic schedule – early planning stages – process frameworks and major product functions are identified
- Detailed schedule – specific software tasks are identified
- Scheduling perspective
  - End date is already established
  - End date is set by organization - established after careful analysis

Project Scheduling Principles

- compartmentalization—define distinct activities, actions and tasks. Decomposing product and process
- interdependency—indicate task interrelationship. Either sequential occurrence of tasks or parallel
- Time allocation —allocate number of work units ; start date and end date w.r.t interdependencies and conduction full-time or part-time
- effort validation—be sure resources are available
- defined responsibilities—people must be assigned
- defined outcomes—each task must have a defined outcome. Outcome is work product
- defined milestones— every task should have a set milestones which are achieved only after review for quality is approved

Effort and Delivery Time



Effort and Delivery Time

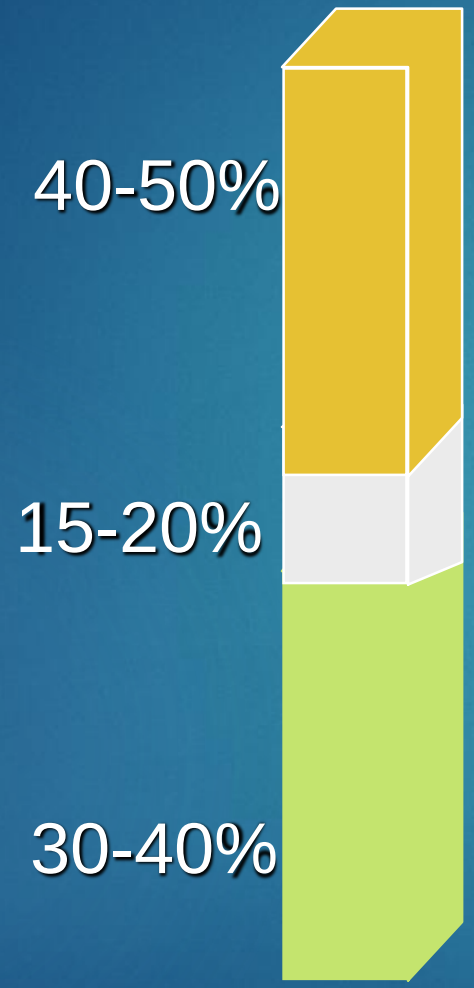
Project Plan: 2 – 3%

Req Analysis: 10 – 25%

Design: 20 – 25%

Coding: 15 – 20%

Testing: 30 – 40%



- "**front end**" activities
  - customer communication
  - analysis
  - design
  - review and modification
- **construction activities**
  - coding or code generation
- **testing and installation**
  - unit, integration
  - white-box, black box
  - regression

Defining Task Sets

- determine type of project
  - Concept development projects
  - New application development projects
  - Application enhancement projects
  - Application maintenance projects
  - Reengineering projects
- assess the degree of rigor required
  - Size of project, no. of users, req stability, appln longevity, ease of communication, maturity of applicable technology, performance constraints, project staff
- identify adaptation criteria
- select appropriate software engineering tasks

Major Task sets

1.1 Concept scoping – determines the overall scope of the project

1.2 Preliminary concept planning – organization's ability to undertake the work implied by the project scope

1.3 Technology risk assessment – evaluate risk associated with technology

1.4 Proof of concept – demonstrates the viability of new technology

1.5 Concept implementation – implements the concept representation in a manner that can be reviewed by a customer

1.6 Customer reaction – feedback from the customer

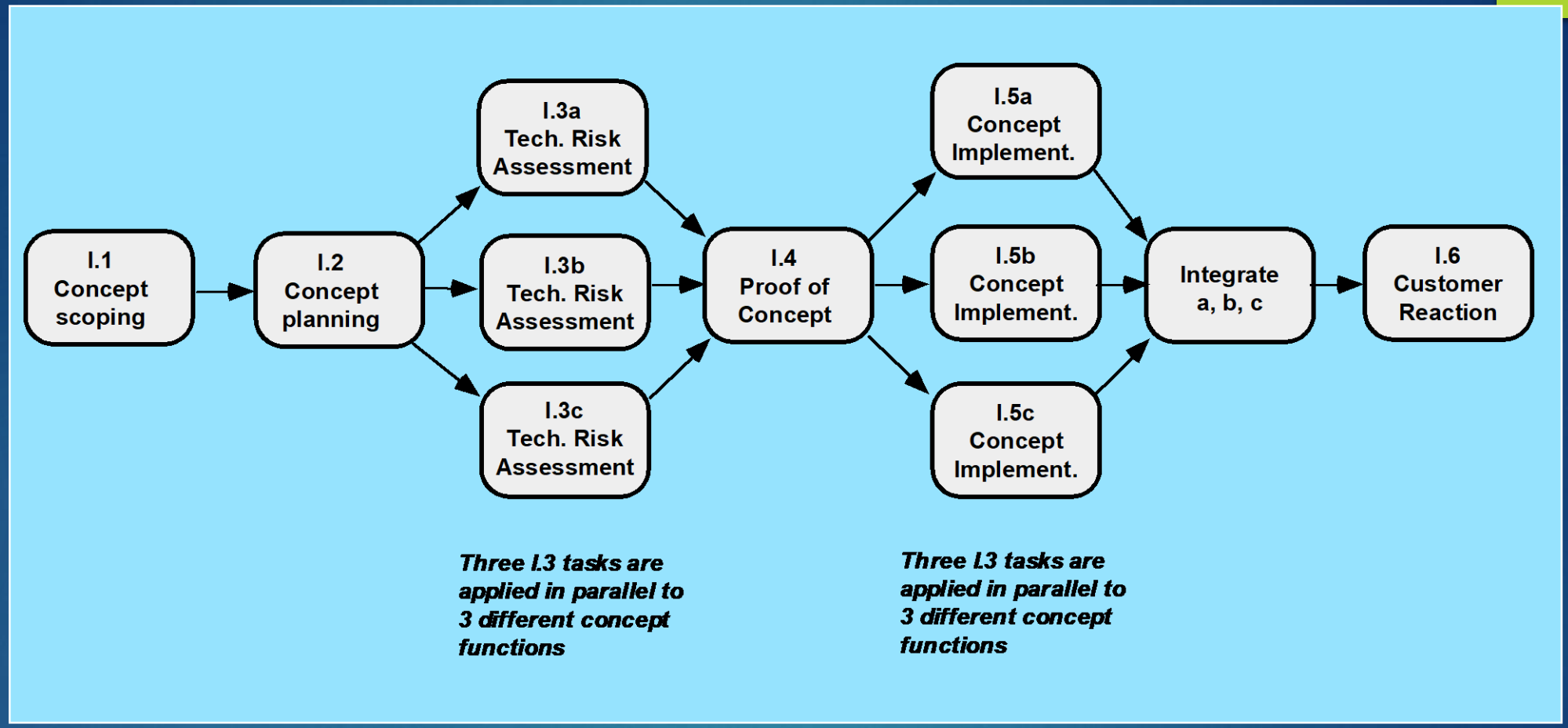
Task Set Refinement

1.1 Concept scoping determines the overall scope of the project.

is refined to

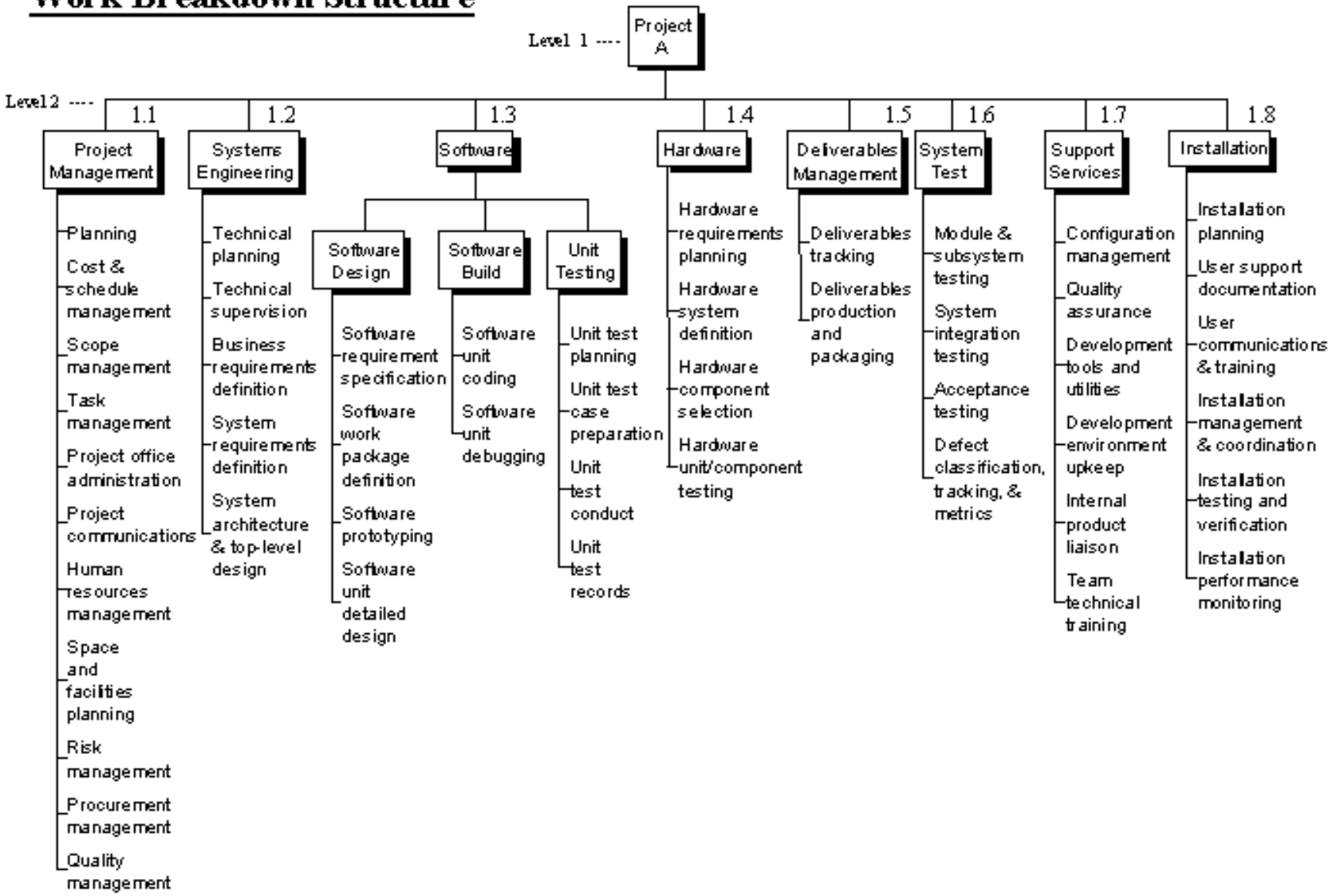
Task definition: Task 1.1 Concept Scoping
1.1.1 Identify need, benefits and potential customers;
1.1.2 Define desired output/control and input events that drive the application;
Begin Task 1.1.2
1.1.2.1 FTR: Review written description of need
FTR indicates that a formal technical review (Chapter 26) is to be conducted.
1.1.2.2 Derive a list of customer visible outputs/inputs
1.1.2.3 FTR: Review outputs/inputs with customer and revise as required;
endtask Task 1.1.2
1.1.3 Define the functionality/behavior for each major function;
Begin Task 1.1.3
1.1.3.1 FTR: Review output and input data objects derived in task 1.1.2;
1.1.3.2 Derive a model of functions/behaviors;
1.1.3.3 FTR: Review functions/behaviors with customer and revise as required;
endtask Task 1.1.3
1.1.4 Isolate those elements of the technology to be implemented in software;
1.1.5 Research availability of existing software;
1.1.6 Define technical feasibility;
1.1.7 Make quick estimate of size;
1.1.8 Create a Scope Definition;
endTask definition: Task 1.1

Define a Task Network



Task network is also called an activity network
Graphic representation of the task flow for a project

Work Breakdown Structure



Kiran Bhowmick

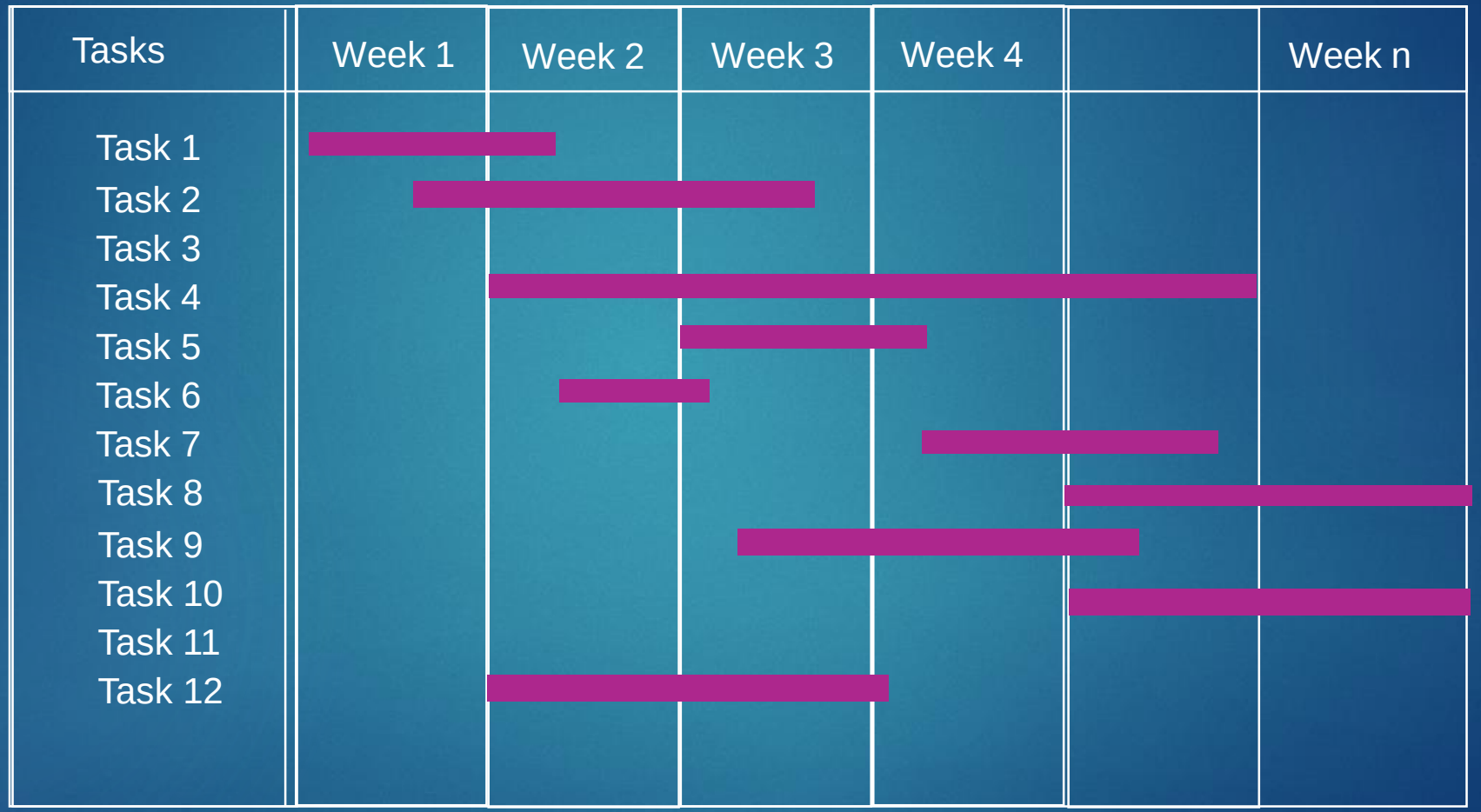
WBS Outline Example

0.0 Retail Web Site
1.0 Project Management
2.0 Requirements Gathering
3.0 Analysis & Design
4.0 Site Software Development
    4.1 HTML Design and Creation
    4.2 Backend Software
        4.2.1 Database Implementation
        4.2.2 Middleware Development
        4.2.3 Security Subsystems
        4.2.4 Catalog Engine
        4.2.5 Transaction Processing
    4.3 Graphics and Interface
    4.4 Content Creation
5.0 Testing and Production

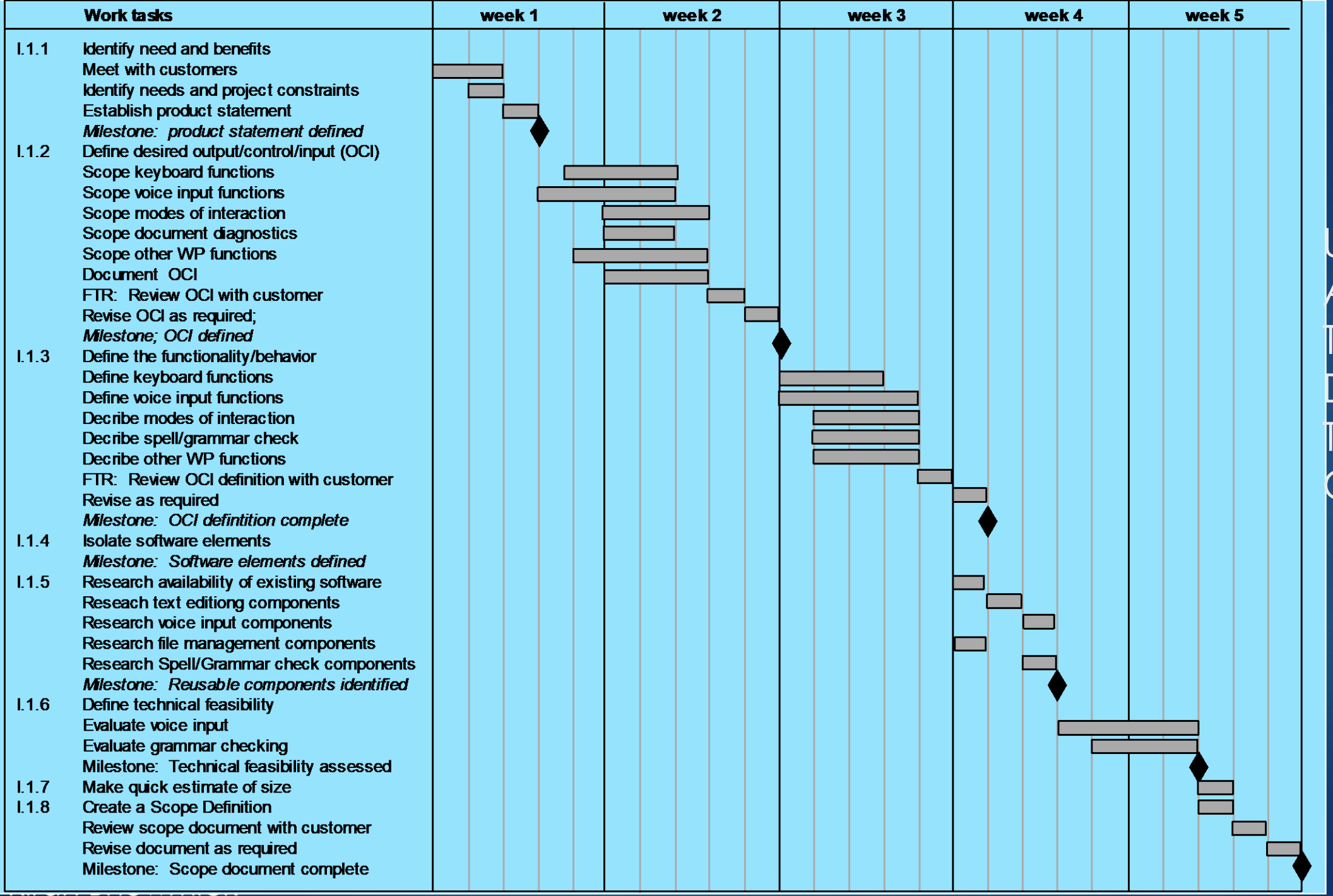
Scheduling

- Project scheduling methods:
  - Program evaluation and review technique (PERT)
  - Critical path method (CPM)
- Quantitative tools that allow the software planner to
  - Determine critical path – the chain of tasks that determines the duration of the project
  - Establish most likely time estimates for individual tasks
  - Calculate boundary times that define a time window for a particular task

Timeline Charts



Use Automated Tools to Derive a Timeline Chart



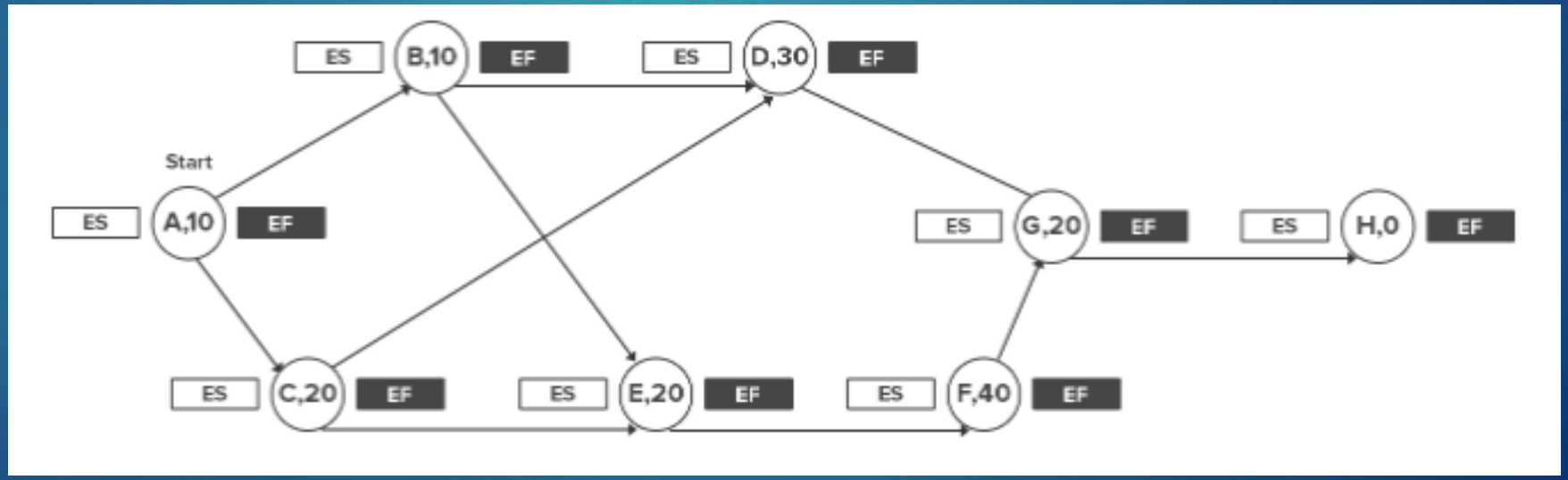
Schedule Tracking

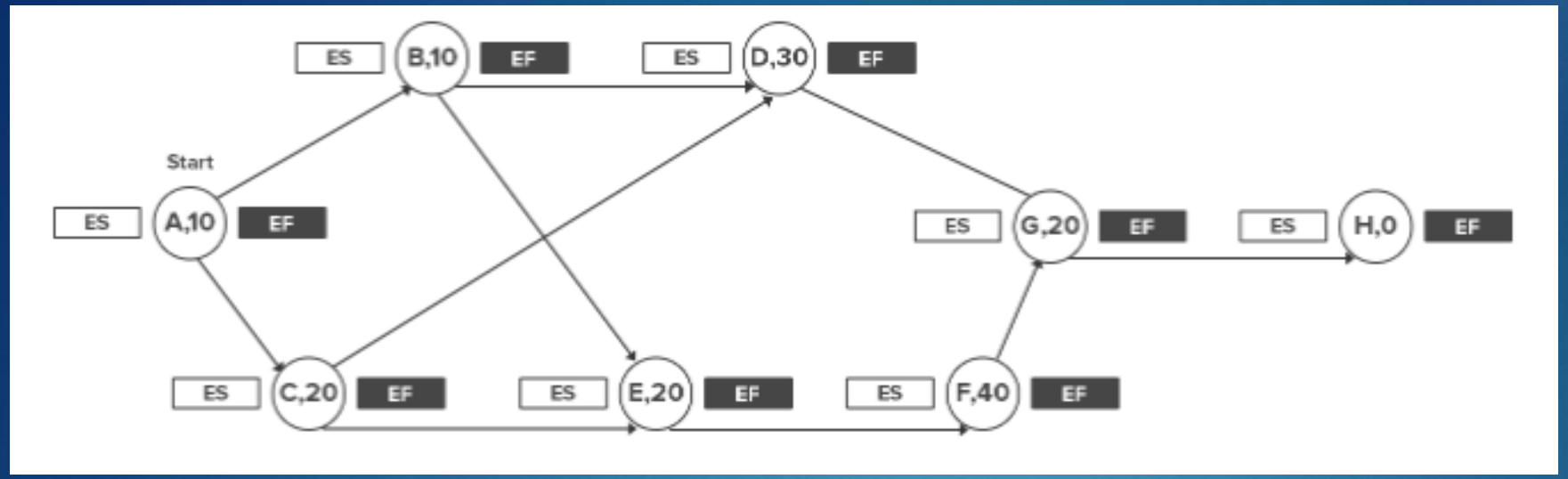
- conduct periodic project status meetings in which each team member reports progress and problems.
- evaluate the results of all reviews conducted throughout the software engineering process.
- determine whether formal project milestones (the diamonds shown in Figure 24.3) have been accomplished by the scheduled date.
- compare actual start-date to planned start-date for each project task listed in the resource table (Figure 24.4).
- meet informally with practitioners to obtain their subjective assessment of progress to date and problems on the horizon.
- use earned value analysis (Section 24.6) to assess progress quantitatively.

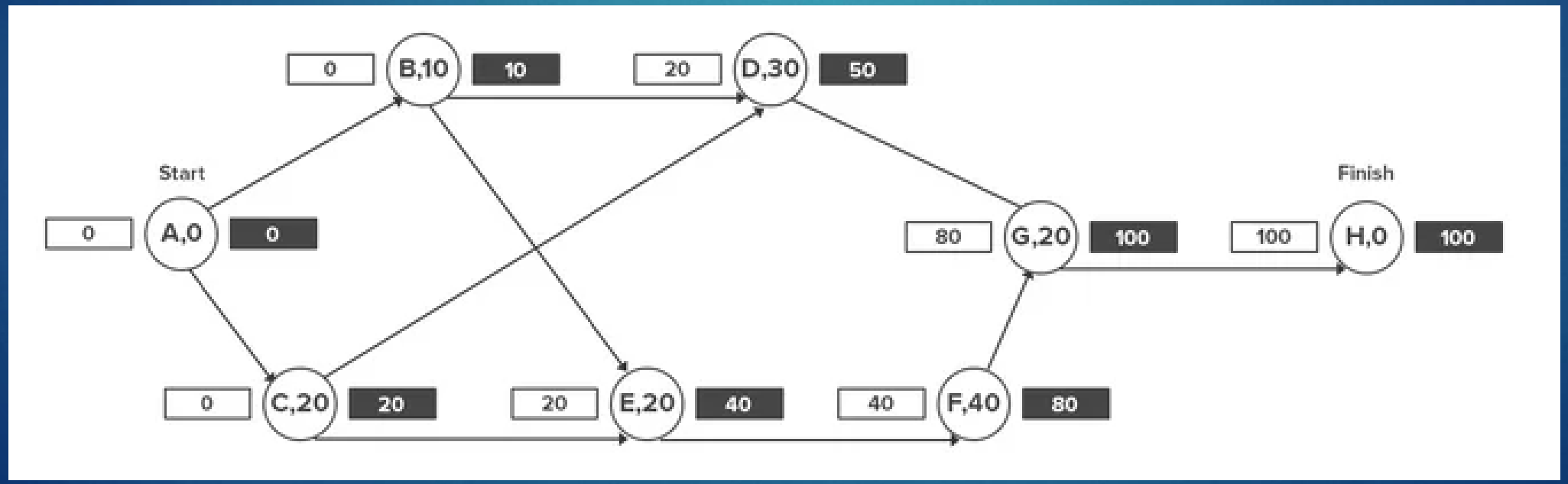
Tracking schedule

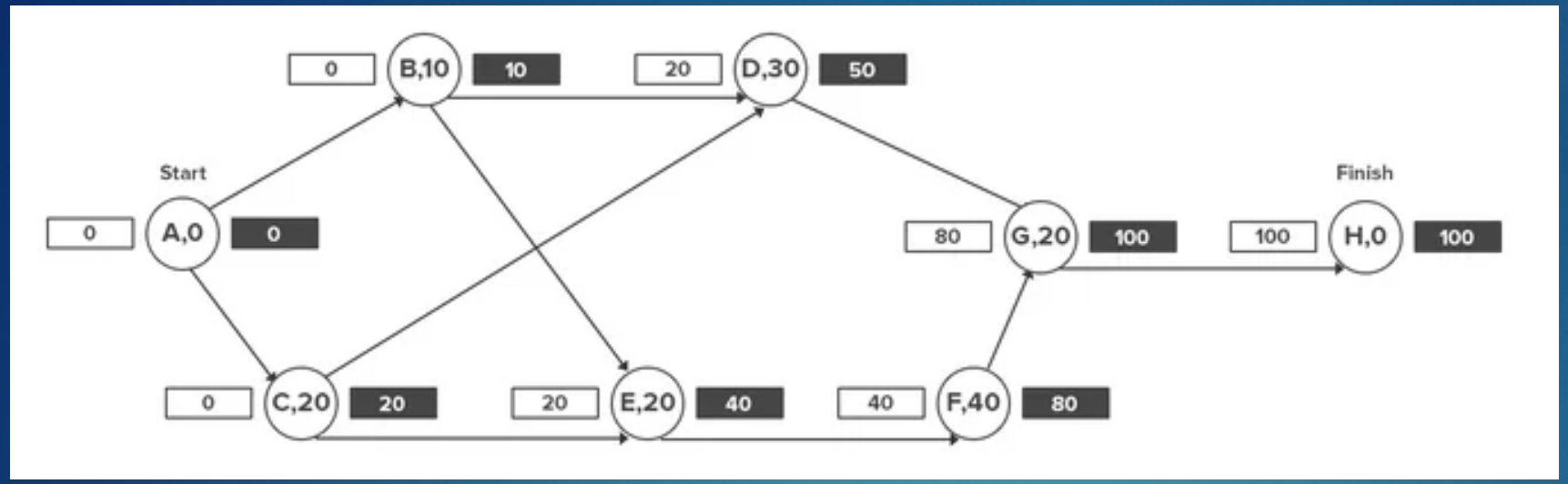
Work tasks	Planned start	Actual start	Planned complete	Actual complete	Assigned person	Effort allocated	Notes
I.1.1 Identify needs and benefits							Scoping will
Meet with customers	wk1, d1	wk1, d1	wk1, d2	wk1, d2	BLS	2 p-d	require more
Identify needs and project constraints	wk1, d2	wk1, d2	wk1, d2	wk1, d2	JPP	1 p-d	effort/time
Establish product statement	wk1, d3	wk1, d3	wk1, d3	wk1, d3	BLS/JPP	1 p-d	
Milestone: Product statement defined	wk1, d3	wk1, d3	wk1, d3	wk1, d3			
I.1.2 Define desired output/control/input (OCI)							
Scope keyboard functions	wk1, d4	wk1, d4	wk2, d2		BLS	1.5 p-d	
Scope voice input functions	wk1, d3	wk1, d3	wk2, d2		JPP	2 p-d	
Scope modes of interaction	wk2, d1		wk2, d3		MLL	1 p-d	
Scope document diagnostics	wk2, d1		wk2, d2		BLS	1.5 p-d	
Scope other WP functions	wk1, d4	wk1, d4	wk2, d3		JPP	2 p-d	
Document OCI	wk2, d1		wk2, d3		MLL	3 p-d	
FTR: Review OCI with customer	wk2, d3		wk2, d3		all	3 p-d	
Revise OCI as required	wk2, d4		wk2, d4		all	3 p-d	
Milestone: OCI defined	wk2, d5		wk2, d5				
I.1.3 Define the function/behavior							

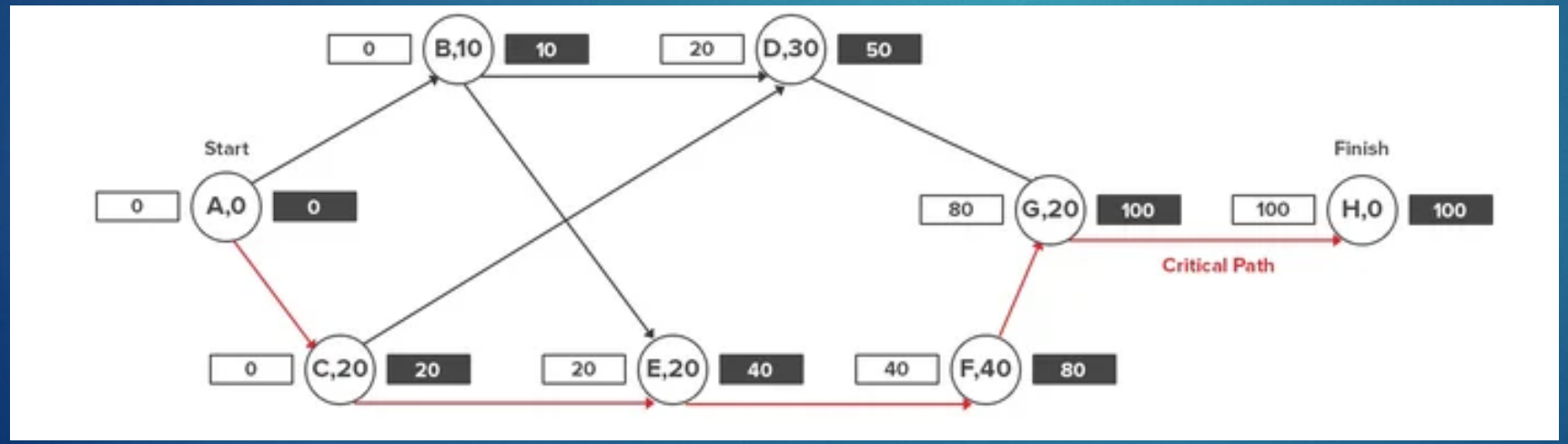
Task ID	Task Description	Task Predecessors	Task Duration (hours)
A	Project start		0
B	Buy materials for A	A	10
C	Buy materials for B	A	20
D	Build A	B, C	30
E	Build B	B, C	20
F	Polish and finish B	E	40
G	Join A and B	D, F	20
H	Project finish	G	0




ES
B,10
EF
ES
D,30
EF
Start
ES
A,10
EF
ES
G,20
EF
ES
H,0
EF
ES
C,20
EF
ES
E,20
EF
ES
F,40
EF


0
B,10
10
20
D,30
50
Start
Finish
0
A,0
0
80
G,20
100
100
H,0
100
0
C,20
20
20
E,20
40
40
F,40
80

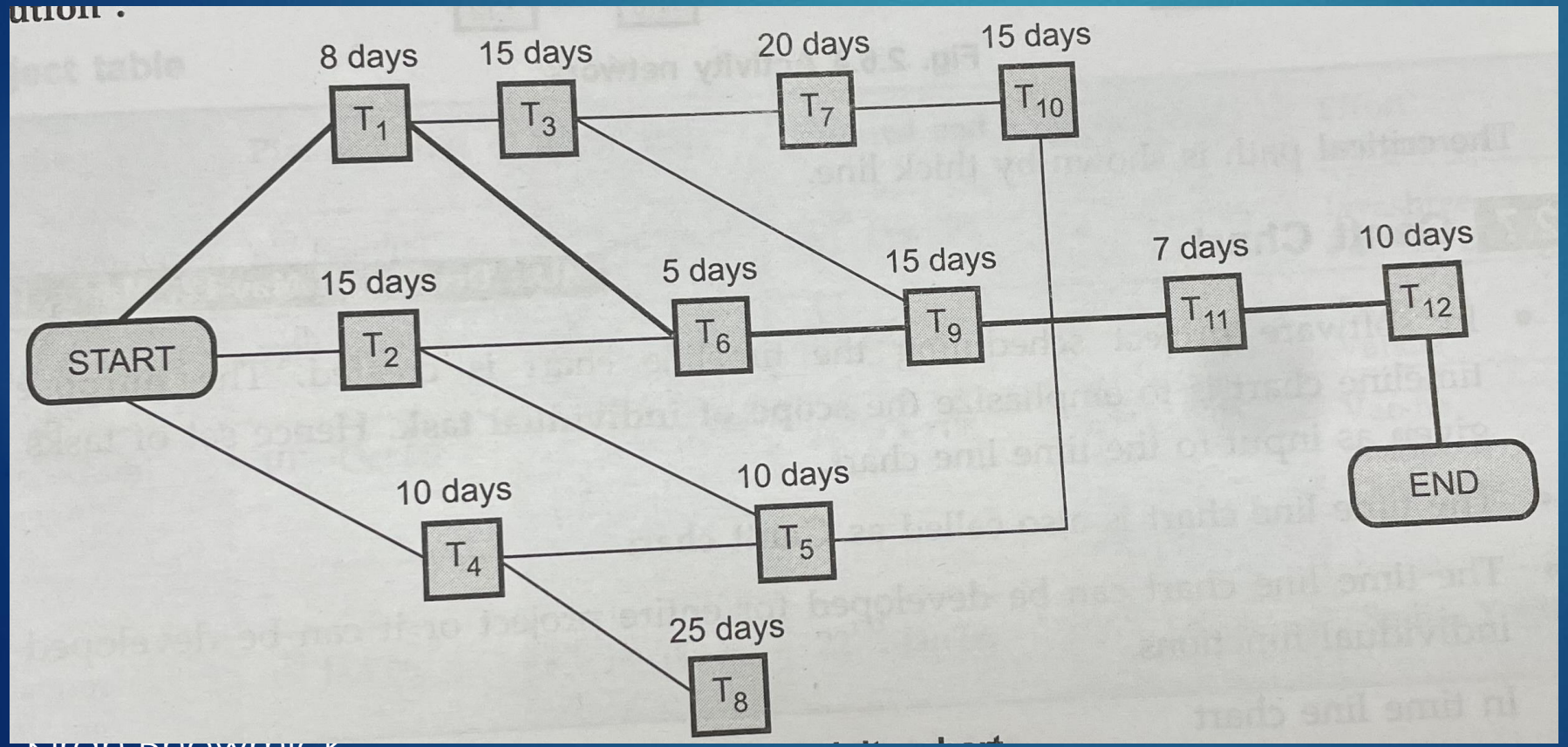




The longest path will be the “critical path”.
The final figure to the right of the last task in the sequence will give you the minimum time the project will take to finish.

Exercise 2.6.2 *For a software project different activities and their durations are listed as below. Draw the activity network and find the critical path.*

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
Duration (in days)	8	15	15	10	10	5	20	25	15	15	7	10
Dependencies	–	–	T_1	–	T_2, T_4	T_1, T_2	T_3	T_4	T_3, T_6	T_5, T_7	T_9	T_{11}



Exercise 2.6.3 *For a software ...*

Draw the activity chart and find the critical path.

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}
Duration in days	10	15	10	20	10	15	20	35	15	5	10	20	35	10	20	10
Dependencies	-	T_1	T_1, T_2	-	-	T_3, T_4	T_3	T_7	T_6	T_5, T_9	T_9	T_{10}	T_3, T_4	T_8, T_9	T_{12}, T_{14}	T_{15}

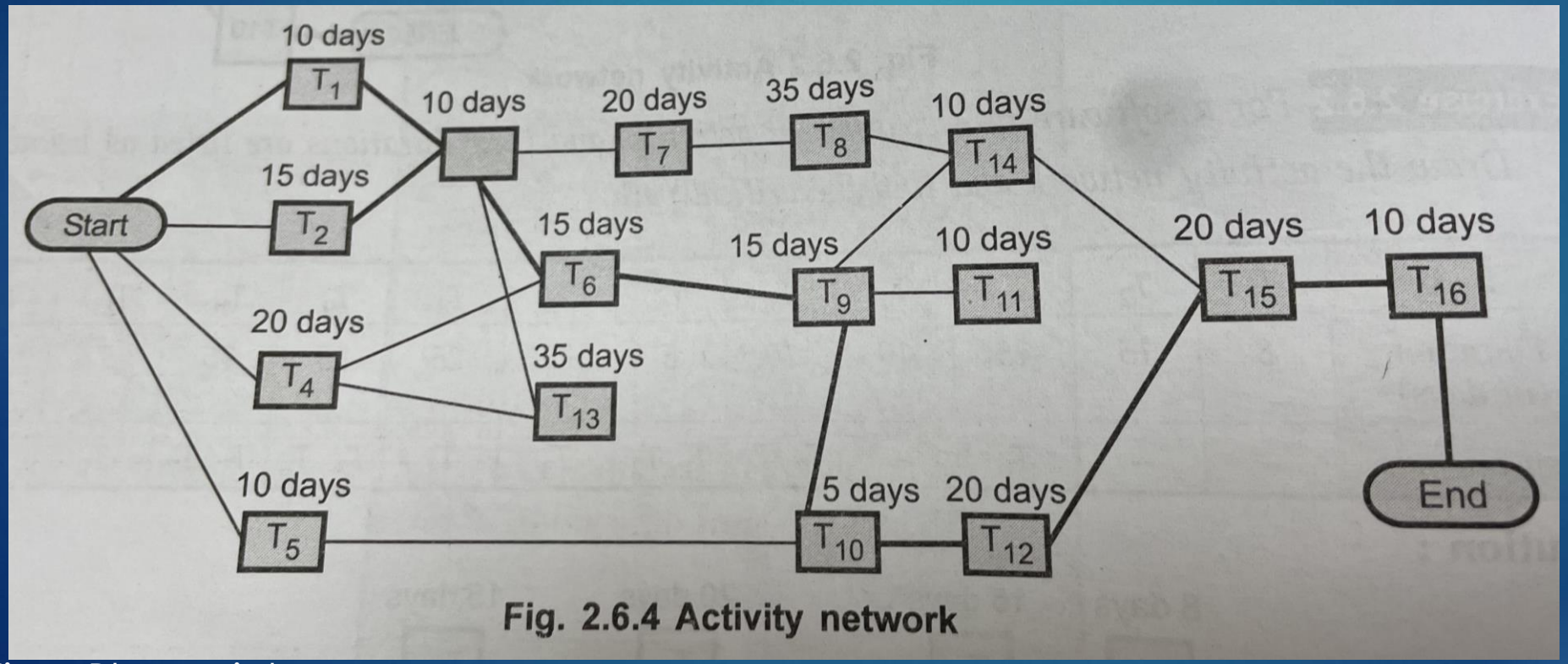


Fig. 2.6.4 Activity network