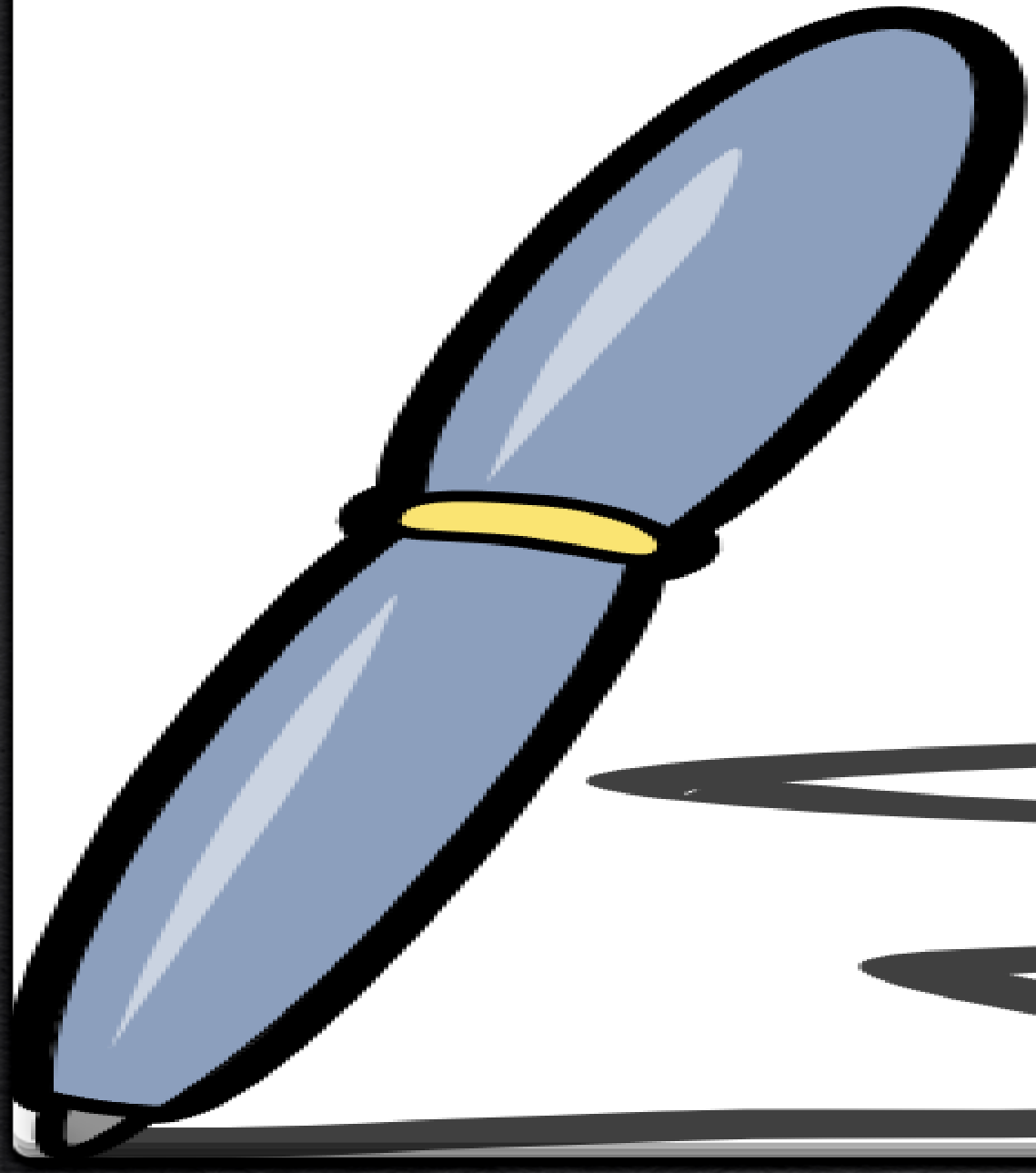
Software project Management

Software Engineering
UNIT-IV

Gurbakash Phonsa



COCOMO

Boehm's model

Introduction

- The COnstructive COst MOdel (COCOMO) is an example of regression models used for estimating software cost and effort.
- The COCOMO Model is the most widely used and accepted of the cost / effort estimation models.
- This model as a size-driven model is highly dependent upon the manager's ability to estimate the size of the software system at an early stage.

Basic COCOMO Model

- Estimates of required effort (measured in Staff-Months **SM/PM** Persons Months)
- Based primarily on your estimate of the software **project's size** (in thousands of Delivered Source Instructions **KDSI/KLOC**)

  $$SM/PM = a * (KDSI)^b$$

- Basic model also presents an equation for estimating the development schedule (Time of Develop **TDEV**) of the project in months.

  $$TDEV = c * (SM)^d$$

Development Mode

- Every project is considered to be developed in one of three modes:
- **Organic Mode.**
- **Semidetached Mode**
- **Embedded Mode**

COCOMO use the same equations (mentioned in above slide) but with different coefficients (*a, b, c, d* in the effort and schedule equations *)* for each development mode.

Therefore before using the COCOMO model we must be able to recognize the development mode of our project.

Development Mode…..conti

- **Organic Mode:** the project is developed in a **familiar, stable environment** and the product is **similar to previously developed** products. The product is relatively small, and require little innovation. An organic mode project is relatively relaxed about the way the software meets its requirements and interface specifications.
- In organic mode value for coefficients are:

  a=2.4

  b=1.05

  c=2.5

  d=0.38

Development Mode…..conti

- **Semidetached Mode:** project's characteristics are intermediate between Organic and Embedded. "Intermediate" may mean either of two things:
  I. An intermediate level of project characteristics.
  II. A mixture of the organic and embedded mode characteristics.
- In an Semidetached mode project
  - Team members all have an intermediate level of experience
  - Team has a wide mixture of experienced and inexperienced people
  - Semidetached mode product generally extends up to 300 KDSI
- In semidetached mode value for coefficients are:

  a=3.0

  b=1.12

  c=2.5

  d=0.35

Development Mode…..conti

Embedded Mode:

- Project is characterized by tight , inflexible constraints and interface requirements.
- Product must operate within a strongly coupled complex of hardware, software, regulations, and operational procedures.
- Project does not generally have the option of negotiating easier software changes and fixes by modifying the requirements and interface specifications.

- In embedded mode value for coefficients are:

  a=3.6

  b=1.2

  c=2.5

  d=0.32

Exercise

A large chemical products company, is planning to develop a new computer program to keep track of raw materials. It will be developed by an in-house team of programmers and analysts who have been developing similar programs for several years. An initial study has determined that the size of the program will be roughly 32,000 delivered source instructions.

Calculate efforts, productivity and time for development.

Function Point

Function Count

Alan Albrecht while working for IBM, recognized the problem in size measurement in the 1970s, and developed a technique (which he called Function Point Analysis), which appeared to be a solution to the size measurement problem.

2.Function Count(Cont.)

The principle of Albrecht's function point analysis (FPA) is that a system is decomposed into functional units.

- Inputs : information entering the system
- Outputs : information leaving the system
- Enquiries : requests for instant access to information
- Internal logical files : information held within the system
- External interface files : information held by other system that is used by the system being analyzed.

2.Function Count(Cont.)

The five functional units are divided in two categories:

(i) Data function types

- Internal Logical Files (ILF): A user identifiable group of logical related data or control information maintained within the system.
- External Interface files (EIF): A user identifiable group of logically related data or control information referenced by the system, but maintained within another system. This means that EIF counted for one system, may be an ILF in another system.

(ii) Transactional function types

- External Input (EI): An EI processes data or control information that comes from outside the system. The EI is an elementary process, which is the smallest unit of activity that is meaningful to the end user in the business.
- External Output (EO): An EO is an elementary process that generate data or control information to be sent outside the system.
- External Inquiry (EQ): An EQ is an elementary process that is made up to an input-output combination that results in data retrieval.

Counting function points

Functional Units	Weighting factors		
	Low	Average	High
External Inputs (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiries (EQ)	3	4	6
Internal logical files (ILF)	7	10	15
External Interface files (EIF)	5	7	10

Table 1 : Functional units with weighting factors

Software Project Planning

Computing function points.

Rate each factor on a scale of 0 to 5.

Number of factors considered (F_i)

1. Does the system require reliable backup and recovery ?
2. Is data communication required ?
3. Are there distributed processing functions ?
4. Is performance critical ?
5. Will the system run in an existing heavily utilized operational environment ?
6. Does the system require on line data entry ?
7. Does the on line data entry require the input transaction to be built over multiple screens or operations ?
8. Are the master files updated on line ?
9. Is the inputs, outputs, files, or inquiries complex ?
10. Is the internal processing complex ?
11. Is the code designed to be reusable ?
12. Are conversion and installation included in the design ?
13. Is the system designed for multiple installations in different organizations ?
14. Is the application designed to facilitate change and ease of use by the user ?

Example: 4.1

Consider a project with the following functional units:

Number of user inputs	= 50
Number of user outputs	= 40
Number of user enquiries	= 35
Number of user files	= 06
Number of external interfaces	= 04

Assume all complexity adjustment factors and weighting factors are average. Compute the function points for the project.

FP

Solution

$$UFP = \sum_{i=1}^{5} \sum_{J=1}^{3} Z_{ij} w_{ij}$$

UFP = 50 x 4 + 40 x 5 + 35 x 4 + 6 x 10 + 4 x 7

= 200 + 200 + 140 + 60 + 28 = 628

CAF = $(0.65 + 0.01 \Sigma F_i)$

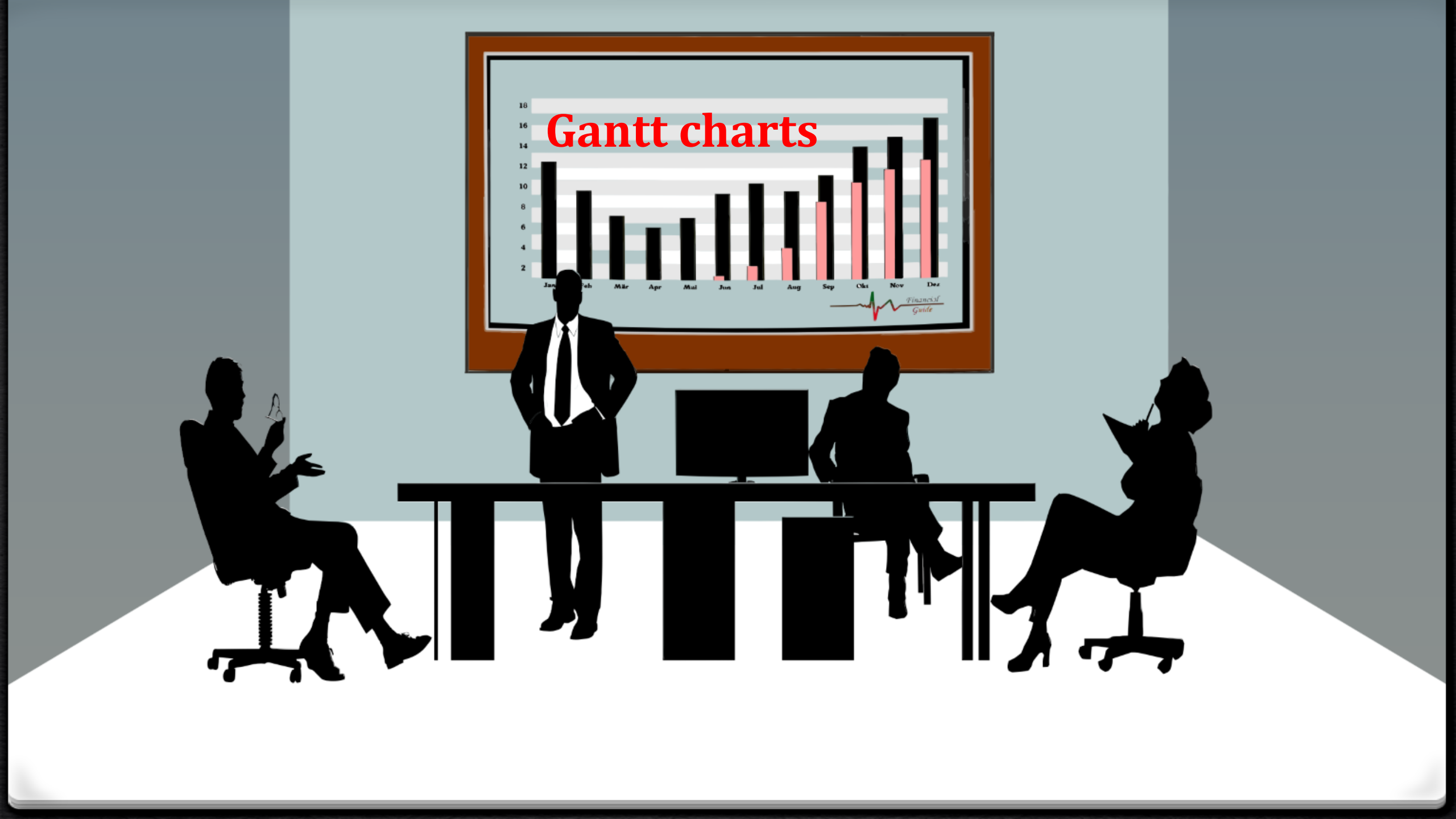
= (0.65 + 0.01 (14 x 3)) = 0.65 + 0.42 = 1.07

FP (Function point) = UFP x CAF

= 628 x 1.07 = 672

CHART

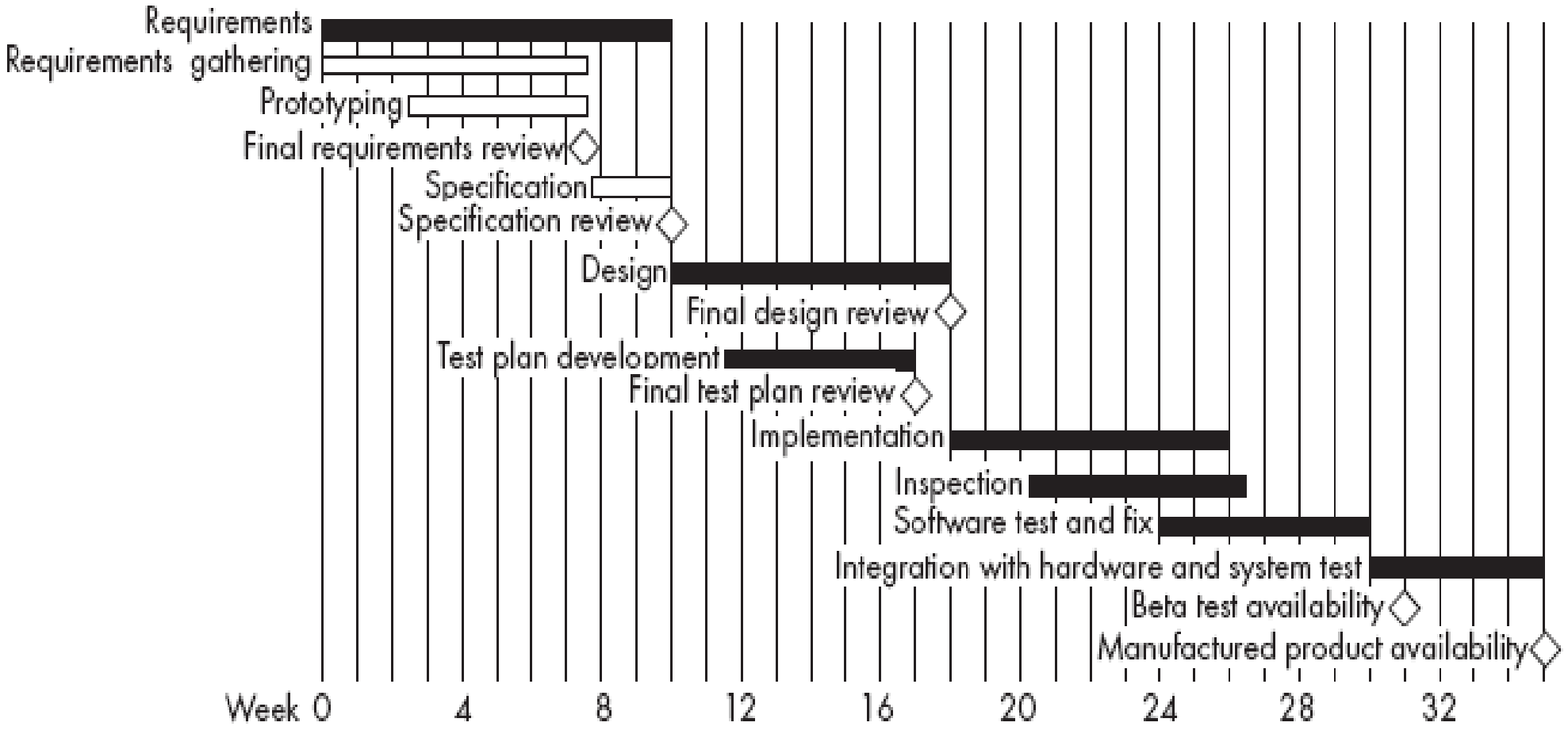
Gantt charts



Gantt charts

- **A Gantt chart is used to graphically present the start and end dates of each software engineering task**
  - One axis shows time.
  - The other axis shows the activities that will be performed.
  - The black bars are the top-level tasks.
  - The white bars are subtasks
  - The diamonds are *milestones:*
    - **Important deadline dates, at which specific events may occur**

Example of a Gantt chart

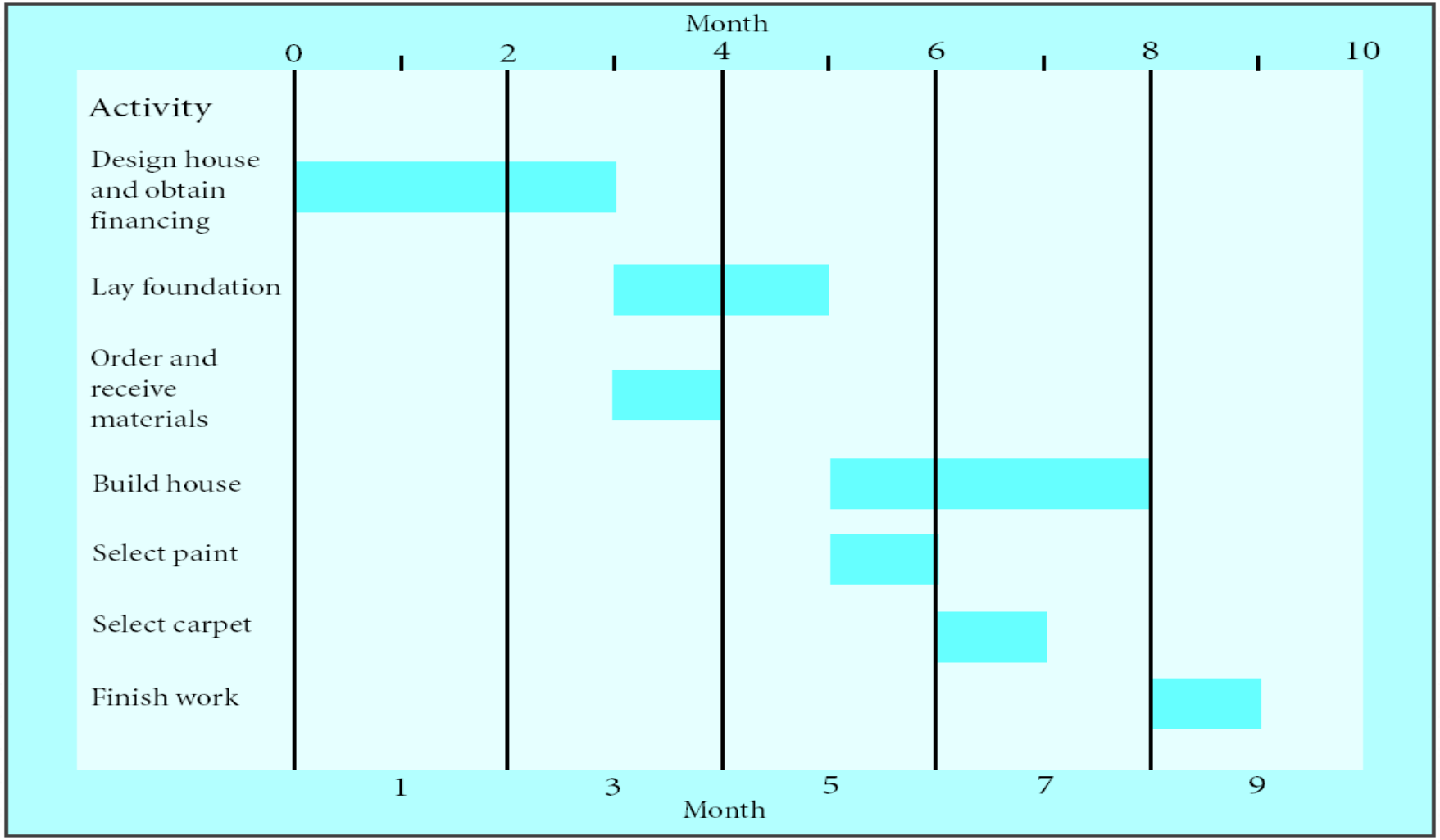


Sequence of Activities of The Project - House Building

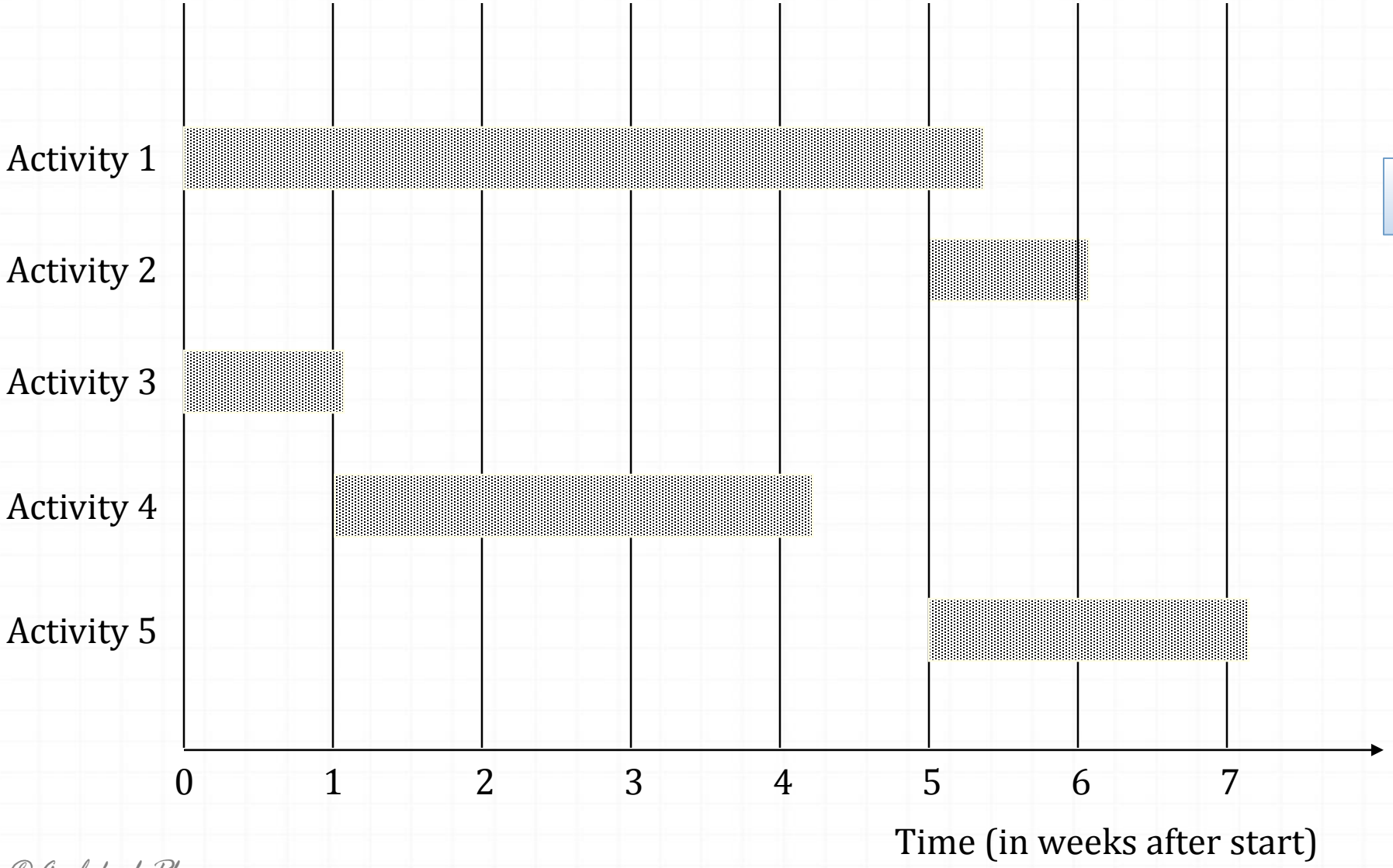
Number	Activity	Predecessor	Duration
1	Design house and obtain financing	--	3 months
2	Lay foundation	1	2 months
3	Order and receive materials	1	1 month
4	Build house	2,3	3 months
5	Select paint	2, 3	1 month
6	Select carper	5	1 month
7	Finish work	4, 6	1 month

Gantt Chart for House Building Project

A Gantt chart



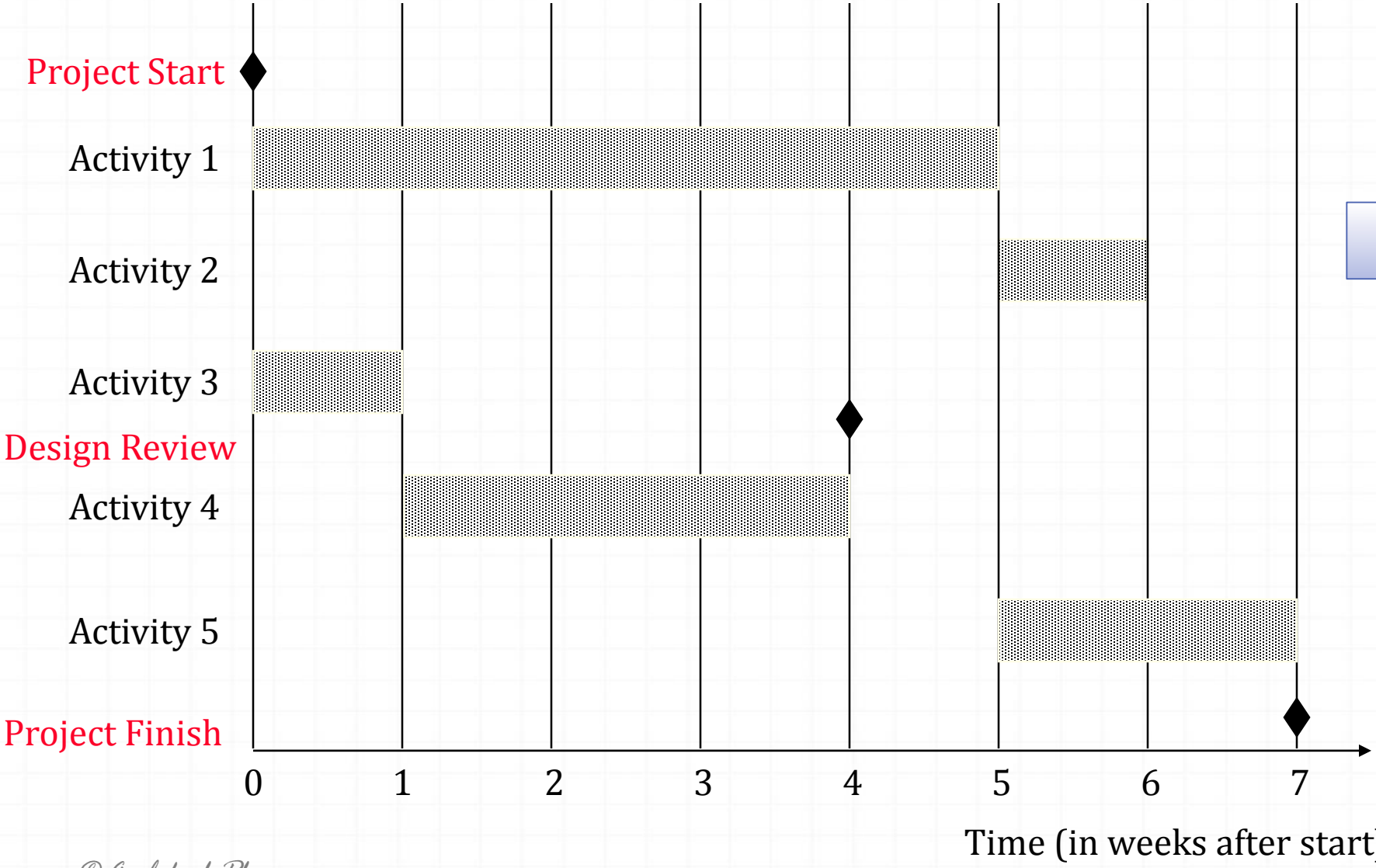
Gantt Chart



Easy to read

Gantt Chart

with milestones

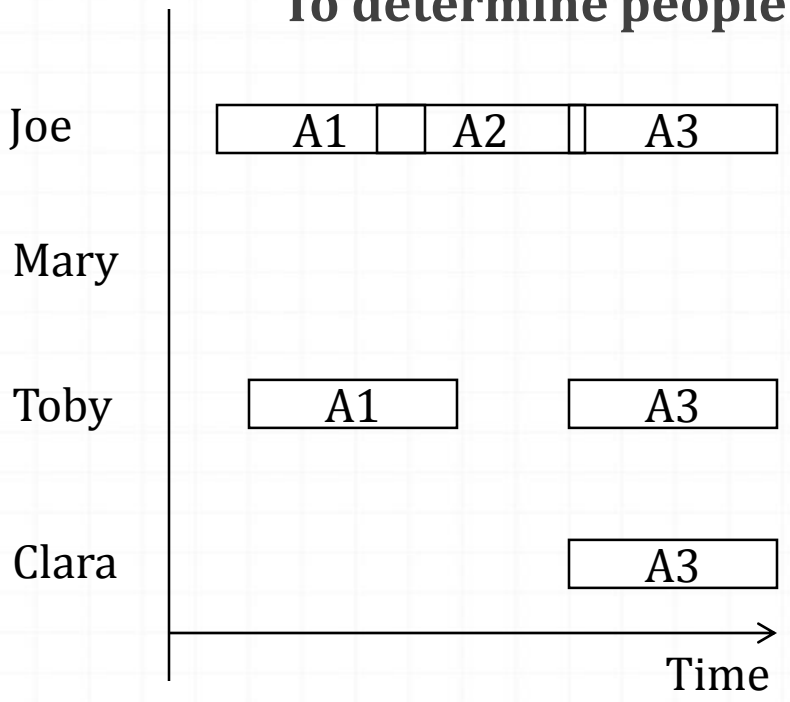


Good for reviews.

Two Types of Gantt Charts

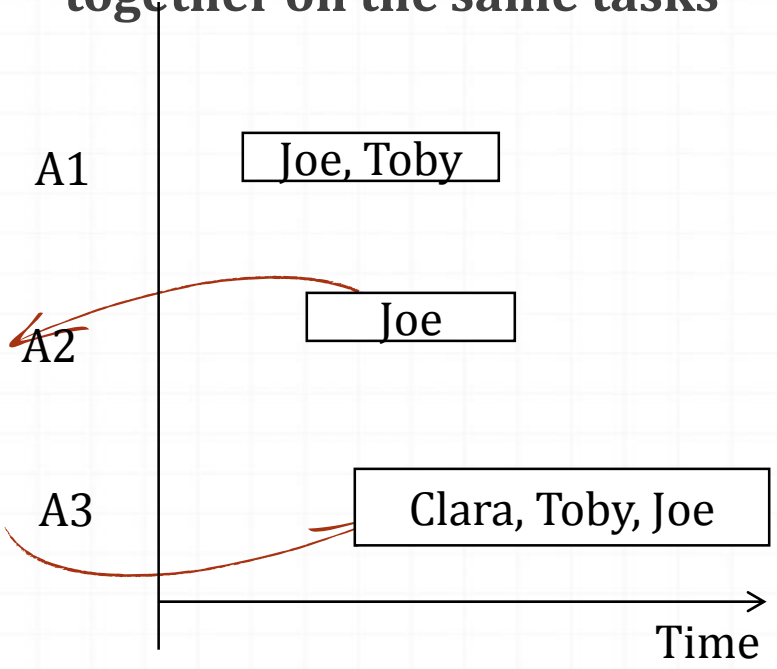
Person-Centered View

To determine people's load



Activity-Centered View

To identify teams working together on the same tasks



Choose one view, stay with it. Usually base the view on the WBS structure Managing Experienced Teams:

1. Person-centered view
2. Managing Beginners: Activity oriented view

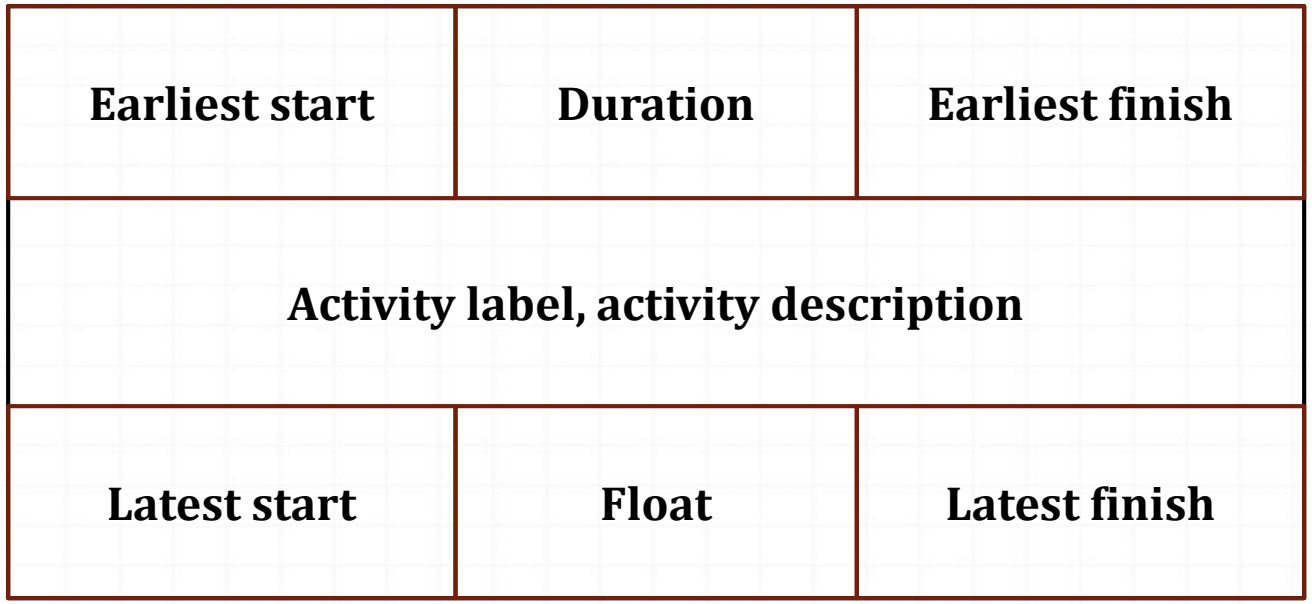
Critical Path Method

Critical Path Method

- Definition: In **CPM** activities are shown as a network of precedence relationships using activity-on-node network construction
  - Single estimate of activity time
  - Deterministic activity times

- **Critical Path:**
  - Is that the sequence of activities and events where there is no "slack" **i.e.. Zero slack**
  - Longest path through a network
  - minimum project completion time

Activity On-Node



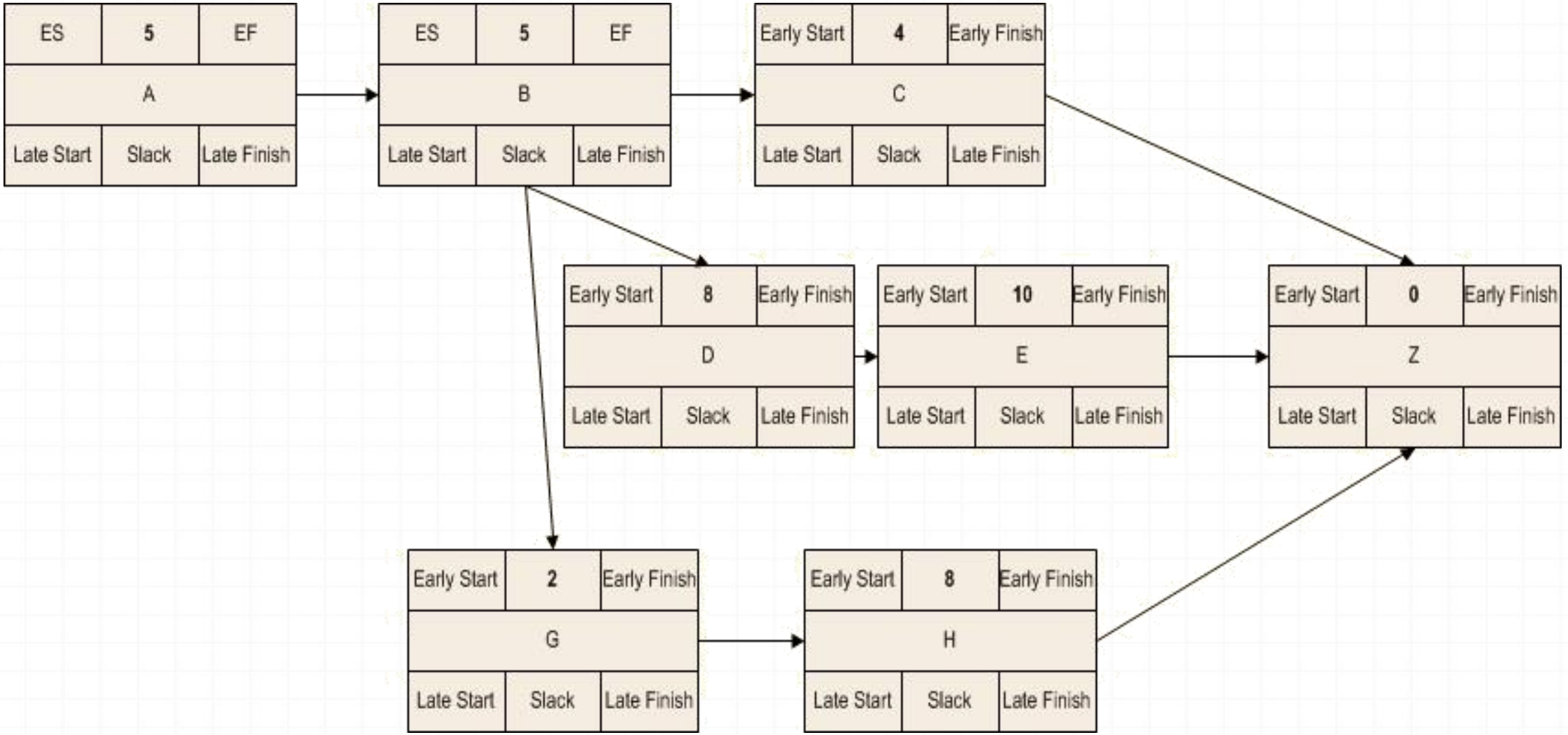
Id.	Activity Name	Duration (days)	Precedents
A	Floating the tender	5	
B	Short listing	5	A
C	Biding	4	B
D	Assigning Contract	8	B
E	Allocation of task	10	D
G	Monitoring	2	B
H	Completing	8	G
Z	System Installation	0	C,E,H

CPM

A **critical path** is the longest path in the network. Each node which falls under critical path has zero or negative float (Slack).

There are 3 steps to calculate CPM:

1. Forward Pass - To calculate the Early Start(ES) and Early Finish(EF) of node.
2. Backward Pass - To calculate Late Start (LS) and Late Finish(LF) of node.
3. Calculate Float and Thus CPM.


ES
5
EF
A
Late Start
Slack
Late Finish
ES
5
EF
B
Late Start
Slack
Late Finish
Early Start
4
Early Finish
C
Late Start
Slack
Late Finish
Early Start
8
Early Finish
D
Late Start
Slack
Late Finish
Early Start
10
Early Finish
E
Late Start
Slack
Late Finish
Early Start
0
Early Finish
Z
Late Start
Slack
Late Finish
Early Start
2
Early Finish
G
Late Start
Slack
Late Finish
Early Start
8
Early Finish
H
Late Start
Slack
Late Finish

The Forward Pass

Node A:

The activity starts on day zero, since A activity duration is for 5 days, the early finish will take Early Start and duration i.e.:

A(EF) = A(ES) + Duration

A (EF)= 0 + 5 = 5

Node B:

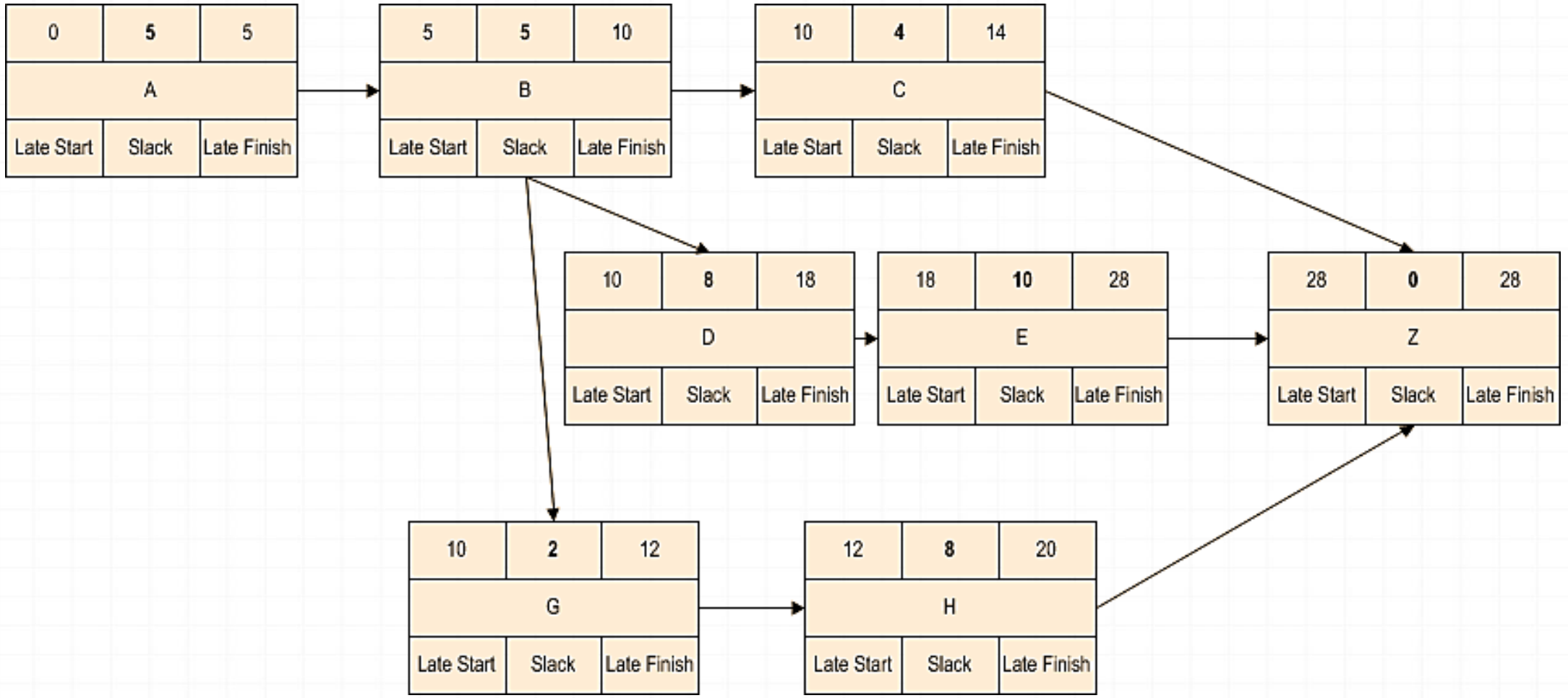
Since there is only one node which precedes activity B. The relationship is F->S. That means that activity B can start only when activity A ends. Hence

B (ES) = A (EF) = 5

B(EF) = B(ES) + Duration = 5+ 5 = 10

Activity C: C(ES) = B(EF) = 10, also C(EF) = C(ES) + Duration = 10+ 4 = 14

Forward pass calculations



Calculation for backward pass

Node H

Node H has only one node preceding it in backward pass (node Z). Hence

H(LF) = Z(LS) = 28

H(LS) = H(LF) – H(Duration) = 28 – 8 = 20

Node E

Node E has only one node preceding it in backward pass (node Z). Hence

E(LF) = Z(LS) = 28

E(LS) = E(LF) – E(Duration) = 28 – 10 = 18

Node C

Node C has only one node preceding it in backward pass (node Z). Hence

C(LF) = C(LS) = 28

C(LS) = C(LF) – C(Duration) = 28 – 4 = 24

More then one forward node

Node B

Since Node B is where most of the activities are merging in backward pass i.e. C, D and G, this is where we need to pay more attention. In backward pass the node B's Latest Finish (LF) would be earliest or all the nodes Late Start i.e.

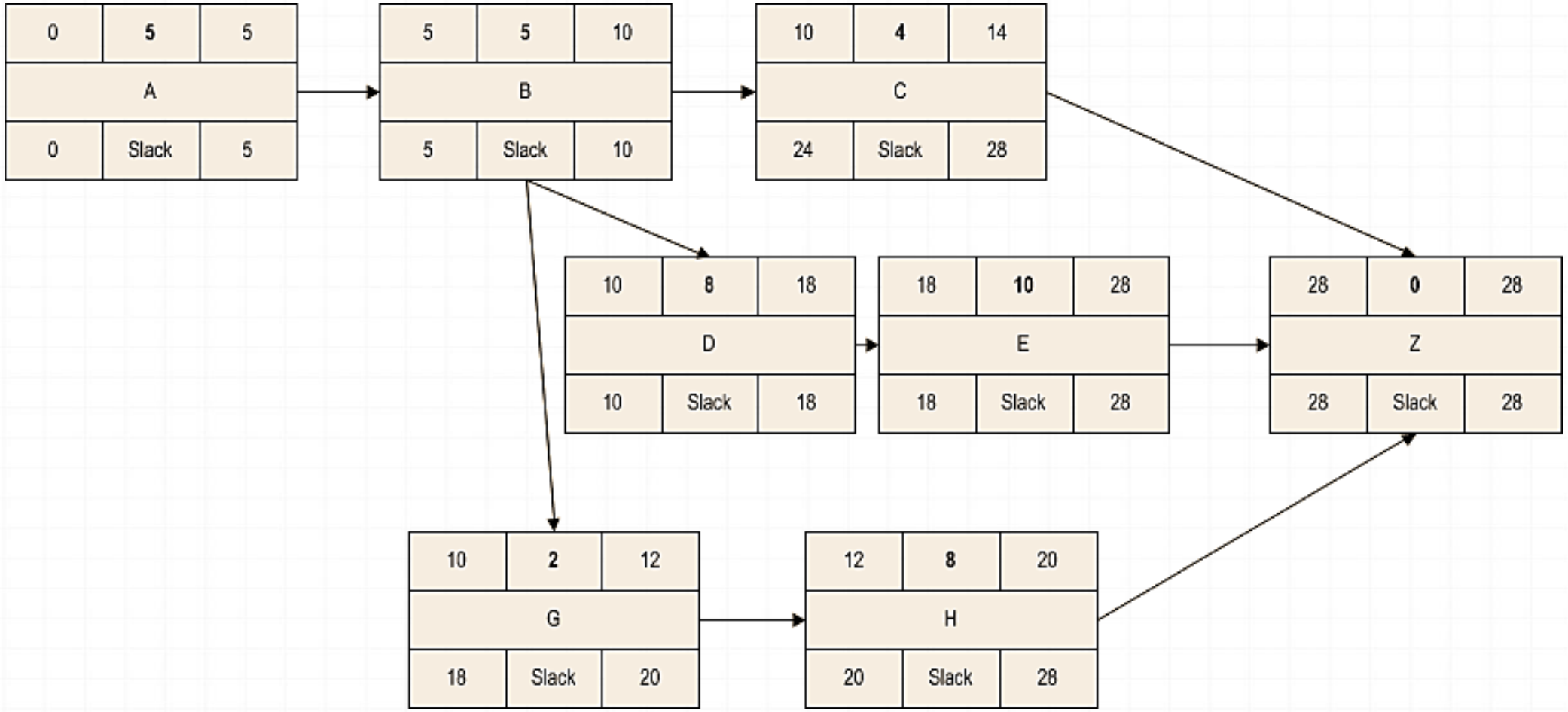
B(LF) = Least | C(LS) or D(LS) or G(LS)|

B(LF) = Least | 24 or 10 or 18|

B(LF) = 10

B(LS) = B(LF) – B(Duration) = 10 -5 = 5

Backward pass calculation



Calculating Total Float or Slack

Total float

Total amount of time that a schedule activity may be delayed from its early start date without delaying the project finish date, or intermediary milestone. It is calculated using:

Activity (ES) – Activity (LS) or activity (EF) Activity(LF) – Both will give you same results.

Free Float

This is an amount of time that a schedule activity can be delayed without delaying the early start of any immediately following schedule activities.

E.g. – For activity C

C (Total Float) = C (LS) - C(ES) = 24 – 10 = 14

C (Free Float) = ES of next activity – C(EF)

C (Free Float) = Z(ES) – C(EF) = 28 -14 = 14

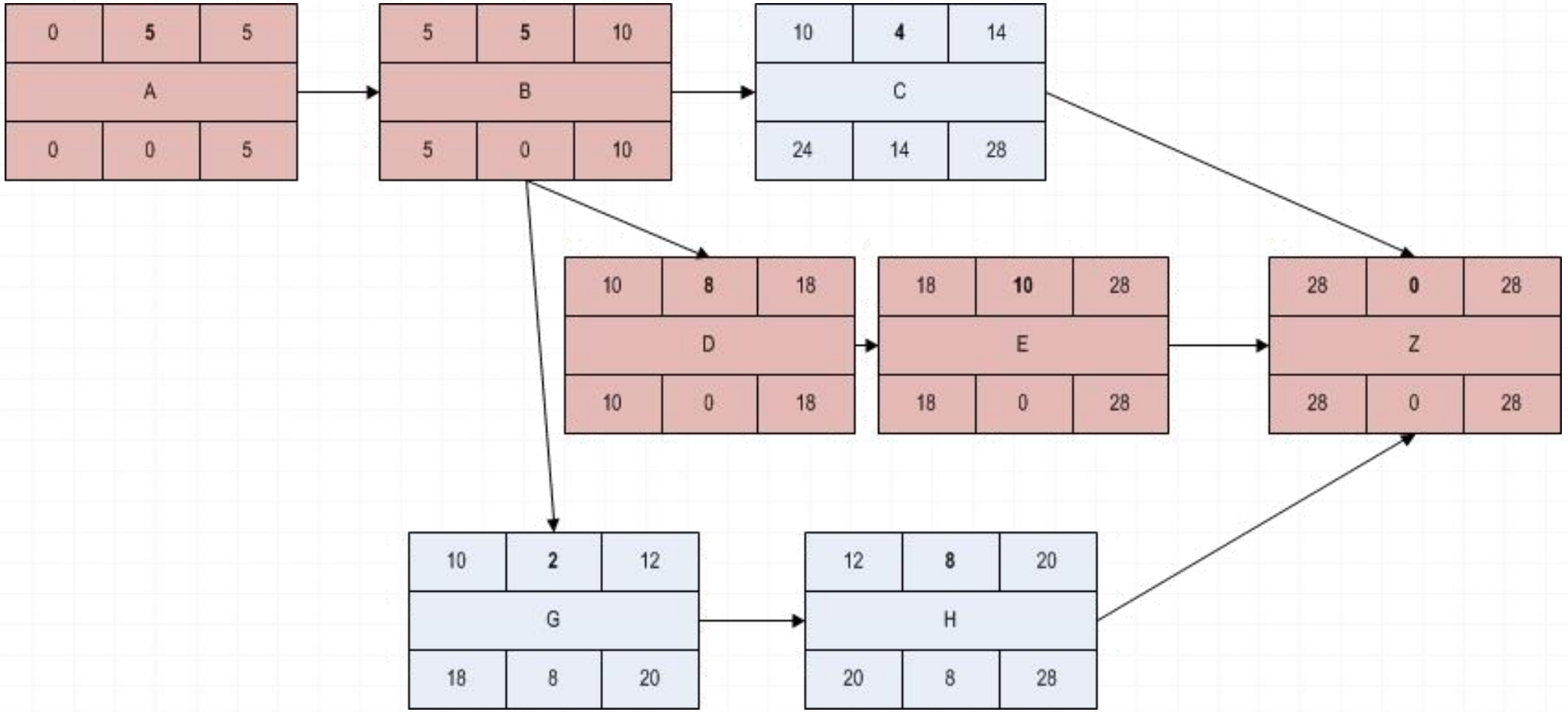
Critical Path

The Critical path

Calculate the Total float for all activities as per the formula.

All nodes which have zero or negative float/slack forms the CRITICAL PATH.

A,B,D,E and Z forms the critical path



Example to construct a CPM

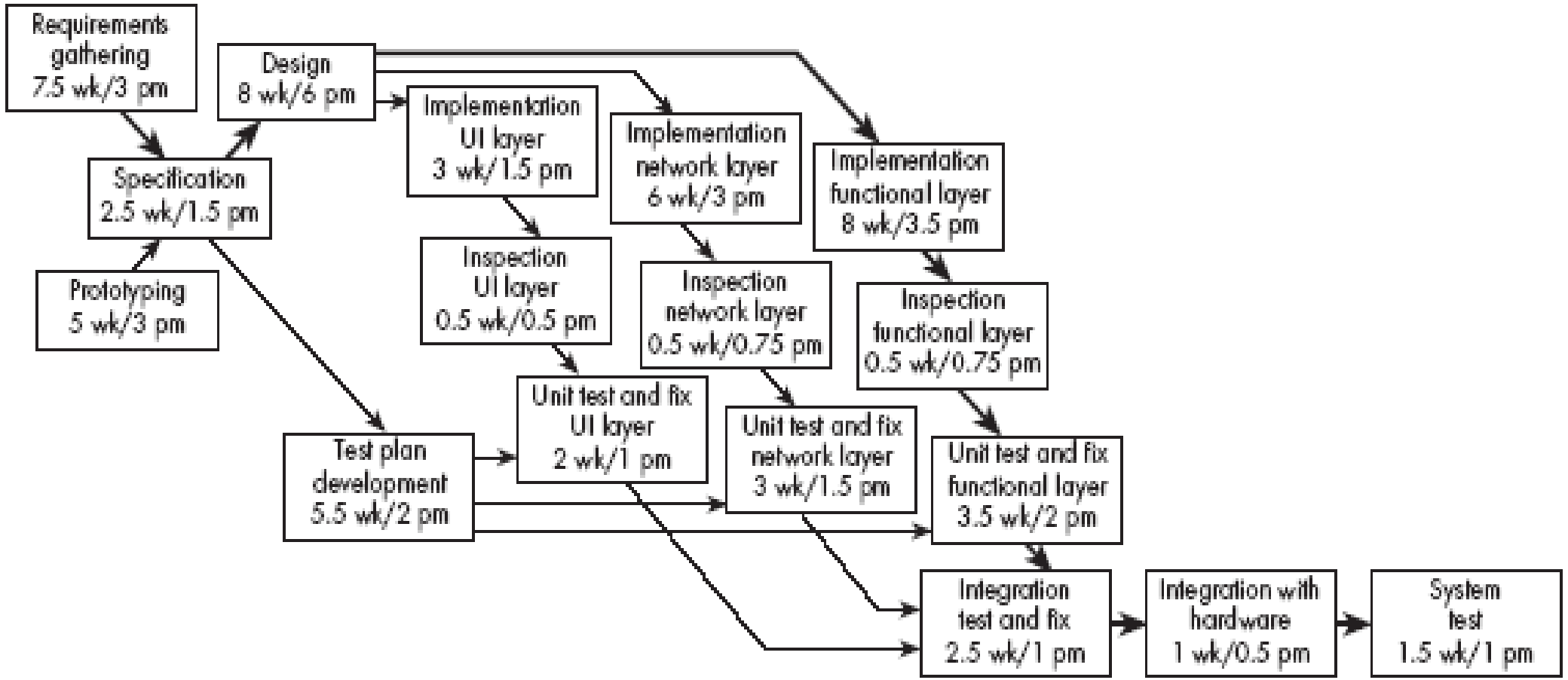
Id.	Activity Name	Duration (weeks)	Precedents
A	Hardware selection	7	
B	Software design	4	
C	Hardware Installation	6	A
D	Coding	4	B
E	Data Preparation	5	B
F	User Documentation	9	
G	User Training	5	E,F
H	System Installation	3	C,D


PERT Chart
18
16
14
12
10
8
6
4
2
Jan
Feb
Mär
Apr
Mai
Jun
Jul
Aug
Sep
Okt
Nov
Dez
Financial
Guide

PERT charts

- A PERT chart shows the sequence in which tasks must be completed.
  - **PERT =** Program Evaluation and Review Technique
  - In each node of a PERT chart, you typically show the elapsed time and effort estimates.
  - The *critical path* indicates the minimum time in which it is possible to complete the project.

Example of a PERT chart



Halstead's Metrics

Halstead's Complexity Metrics

- LOC - a function of complexity
- Language and programmer dependent
- Halstead's Software Science (entropy measures)
  - n_1 - number of distinct operators
  - n_2 - number of distinct operands
  - N_1 - total number of operators
  - N_2 - total number of operands

Example

```
if (k < 2)
{
  if (k > 3)
    x = x*k;
}
```

- Distinct operators: if ( ) { } > < = * ;
- Distinct operands: k 2 3 x
- $n_1 = 10$
- $n_2 = 4$
- $N_1 = 13$
- $N_2 = 7$

Halstead's Metrics

- Amenable to experimental verification [1970s]

- Program length: $N = N_1 + N_2$
- Program vocabulary: $n = n_1 + n_2$
- Estimated length: $\hat{N} = n_1 \log_2 n_1 + n_2 \log_2 n_2$
  - Close estimate of length for well structured programs

$\hat{N}$

Program Complexity

- Volume: $V = N \log_2 n$
  - Number of bits to provide a unique designator for each of the n items in the program vocabulary.
- Difficulty

$$D = \frac{n_1}{2} \times \frac{N_2}{n_2}$$

- Program effort: E=D*V
  - This is a good measure of program understandability