

# Estimation, Scheduling & Tracking



DASS – Spring 2025

# Project Management

---

- ▶ Two ~~over-arching~~ inter-dependent aspects of software projects
  - ▶ Process
  - ▶ Project Management

Main responsibilities of a project manager are

- ▶ Project planning
- ▶ Project monitoring and control

(Major activities: Software Estimation, Scheduling and Tracking)

---

Estimate how long will it take for you to  
get to hostel from class today !



# Estimations

---

- ▶ On what basis did you estimate?
  - ▶ Experience right? (History matters...)
  - ▶ Likely as an “average” probability

Remember, an “exact estimate” is an oxymoron

# Can you estimate these?

---

1. Surface temperature of the sun (in degrees C)
2. Latitude of Hyderabad (in degrees)
3. Surface area of Asia (in km<sup>2</sup>)
4. Birth date of Alexander The Great (year)
5. Global revenue of “Titanic” (in \$)
6. Length of the Pacific coastline (Ca, Or, Wa) (in km)
7. Weight of the largest whale (in tonnes)



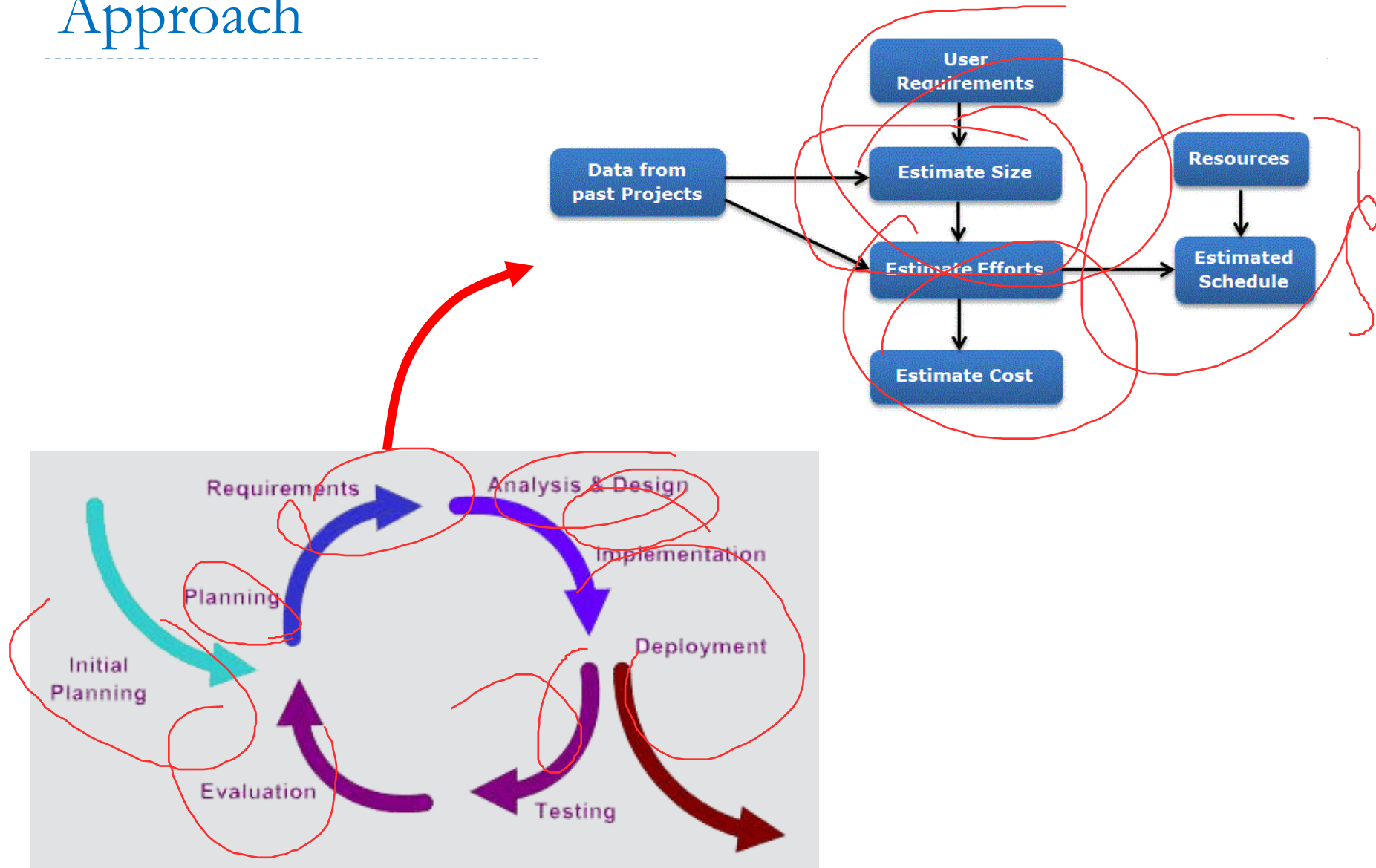
---

# 1

Always give a range  
Never give them a number



# Approach



---

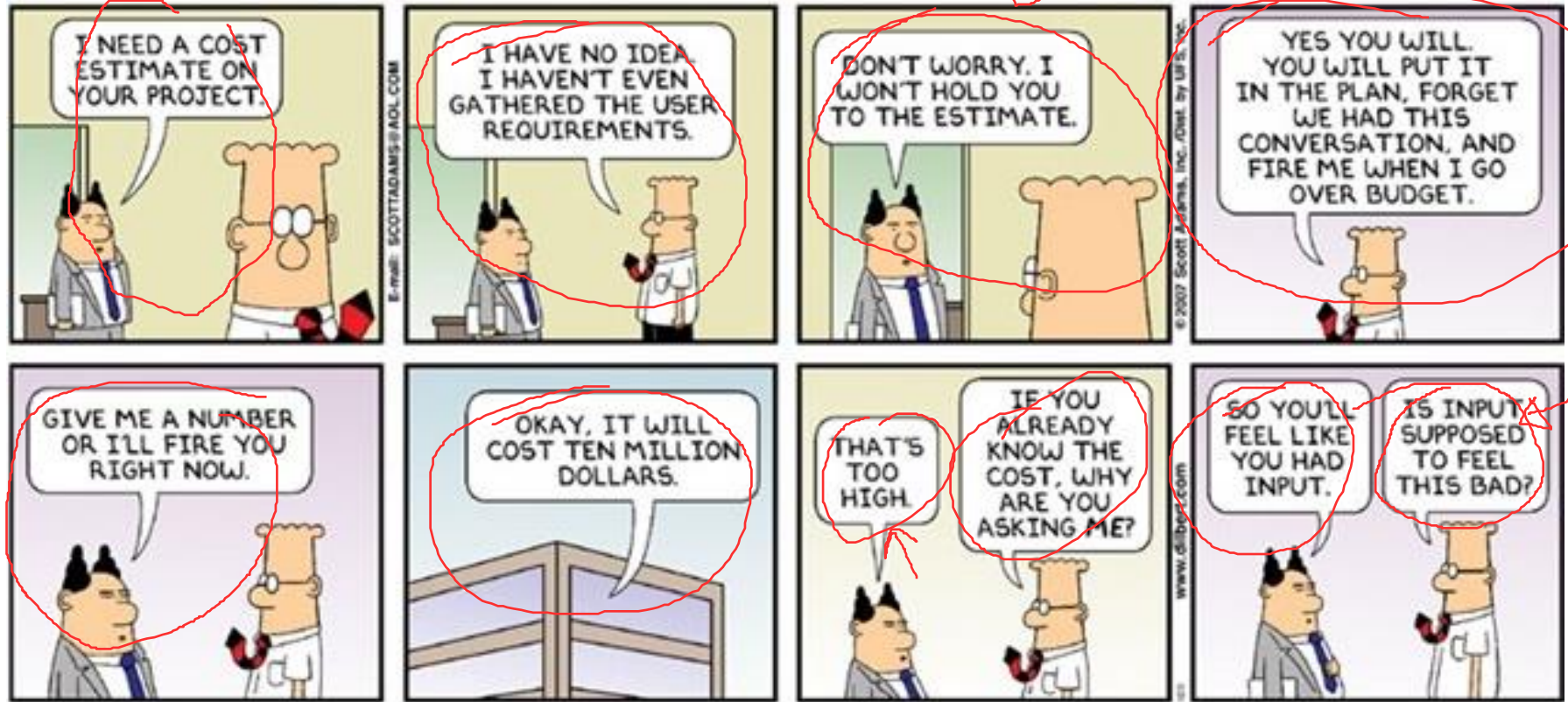
#2

Always ask what the estimate will  
be used for





# Requirements is key



© Scott Adams, Inc./Dist. by UFS, Inc.

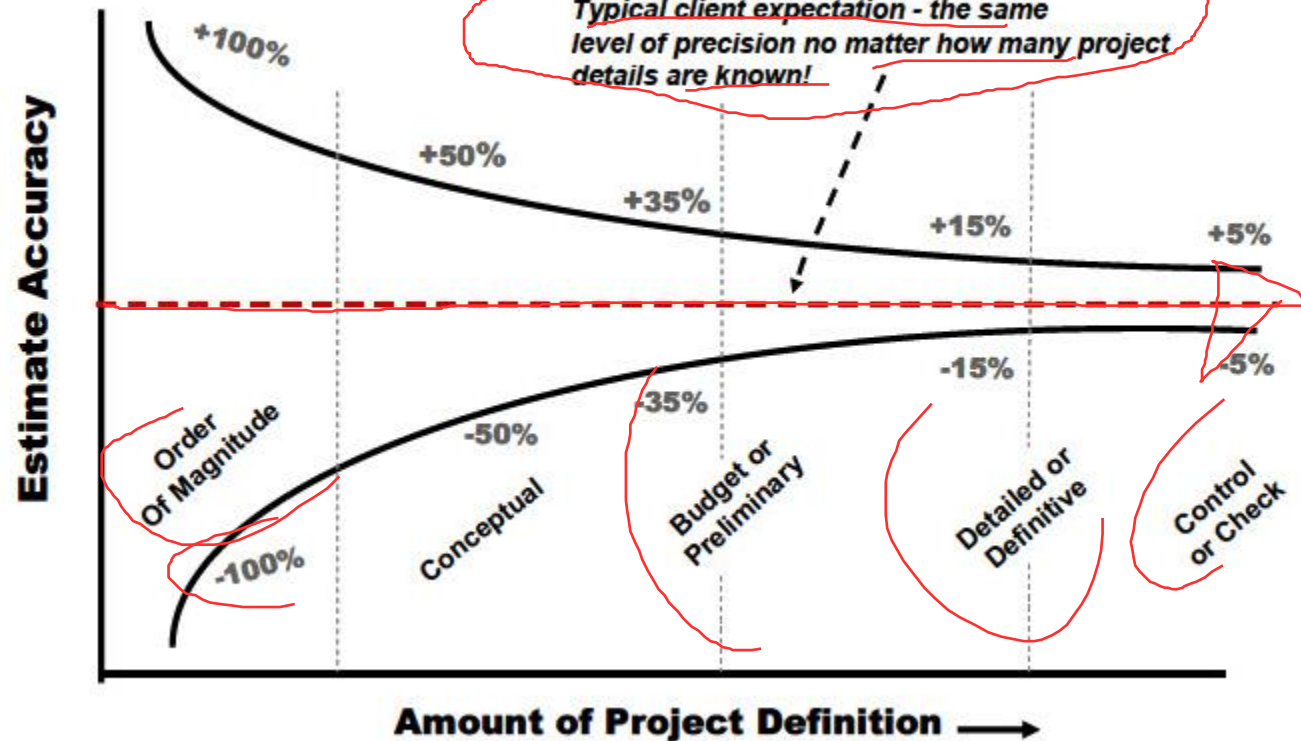
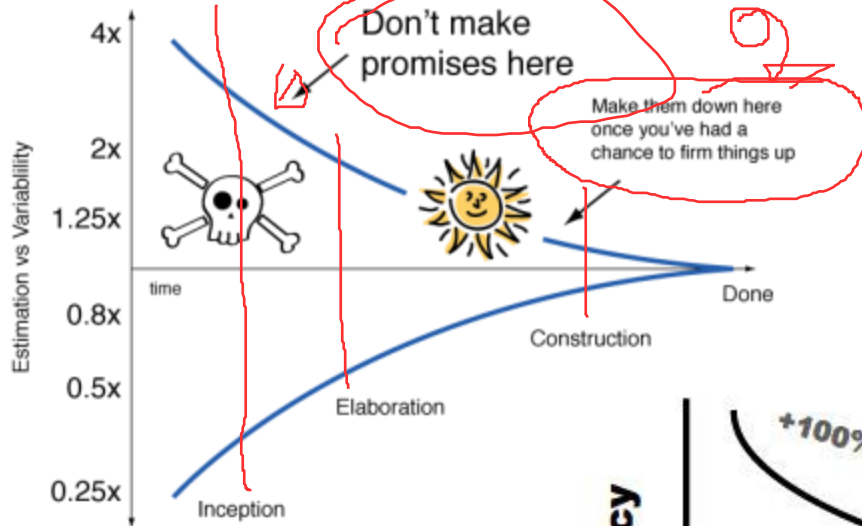
---

#3

Estimation != Commitment



# Iteratively increasing clarity



---

#4

First try to measure, count and  
compute

Estimate only when necessary



# Reality

## Estimation

### Inexperienced Developer



### Experienced Developer



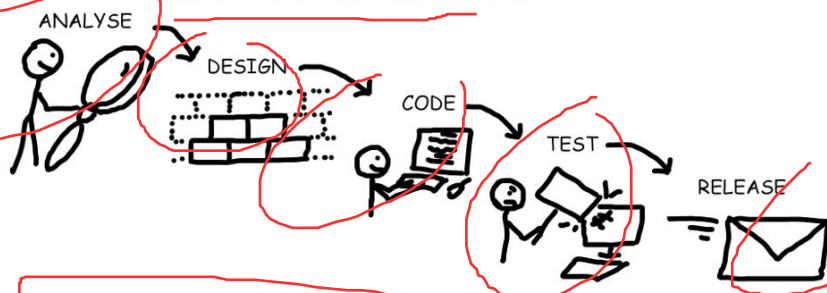
### Inexperienced PM



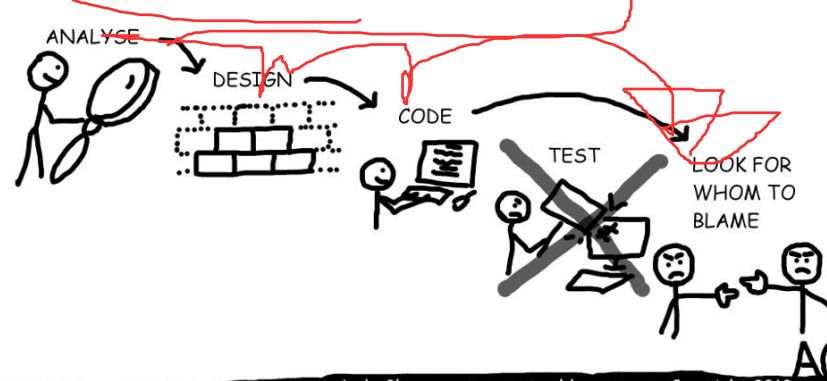
### Experienced PM



### THE WATERFALL SDLC in THEORY



### THE WATERFALL SDLC in PRACTICE



### AND THEN SALES COMES ALONG:



---

#5

Aggregate independent estimates

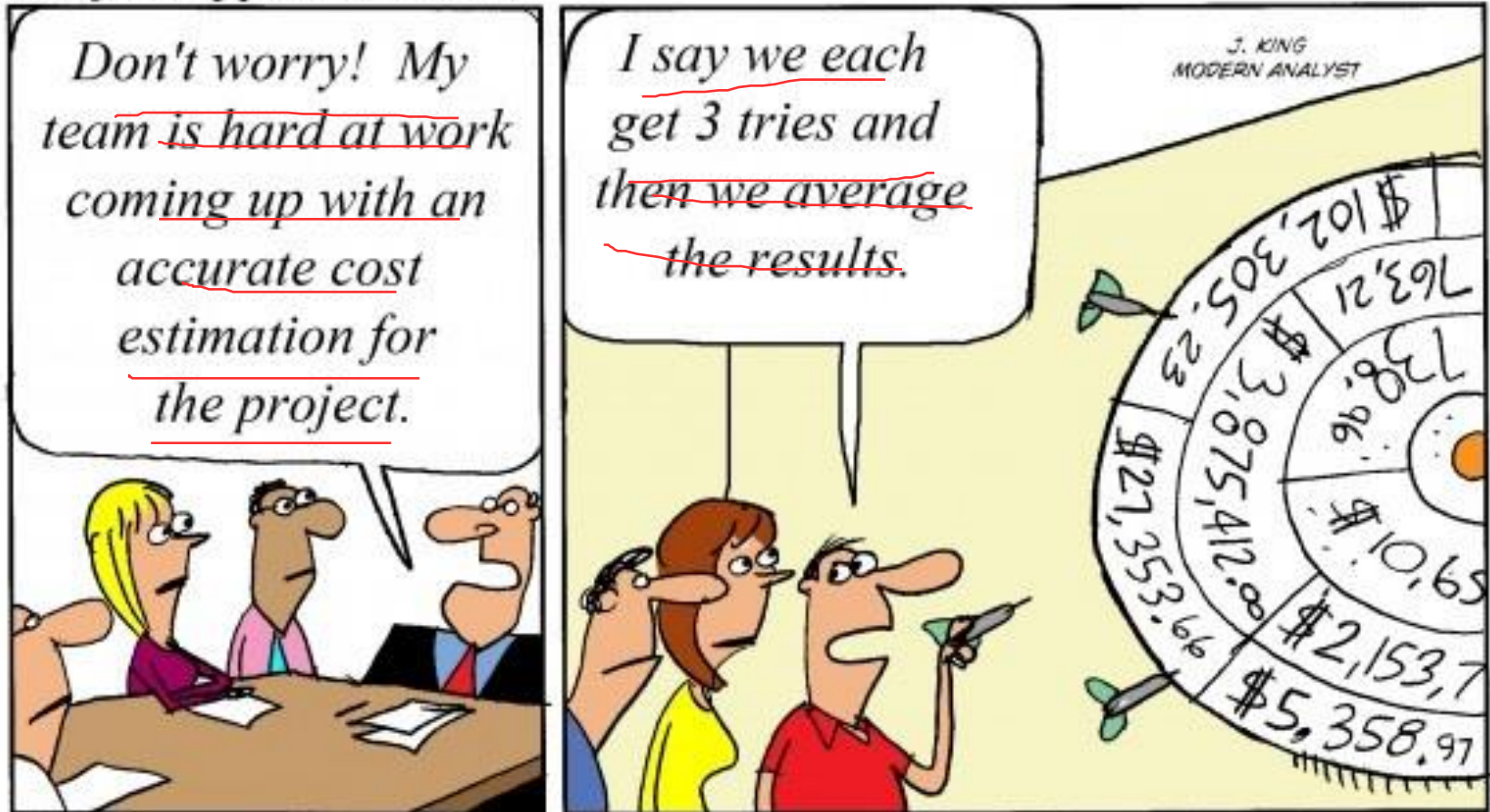
“Wisdom of the Crowds”





# Team effort

## Project Approval Board



# The law of large numbers (or: statistics is on our side, for once)

---

- ▶ If we estimate with an error of  $x\%$
- ▶ The estimate of each scope item will have an error of  $x\%$
- ▶ But...
- ▶ Some items will be over-estimated, others under-estimated (maybe....)

=> The error on the total estimate is likely  $< x\%$



# Estimation Methodologies

---

- ▶ Top-down
- ▶ Bottom-up
- ▶ Analogy
- ▶ Expert Judgment
- ▶ Priced to Win (request for quote – RFQ)
- ▶ Parametric or Algorithmic Method
  - ▶ Using formulas and equations

# Function Point Analysis (FPA)

- ▶ Software size measured by number & complexity of functions it performs
- ▶ More methodical than LOC counts
- ▶ House analogy
  - ▶ House's Square Feet  $\sim$  Software LOC
  - ▶ # Bedrooms & Baths  $\sim$  Function points
  - ▶ Former is size only, latter is size & function
- ▶ Five basic steps

# Function Point Analysis - Process

---

- ▶ 1. Count # of business functions per category
  - ▶ Categories: **outputs, inputs, DB inquiries, files or data structures, and interfaces**
- ▶ 2. Establish Complexity Factor for each and apply
  - ▶ Low, Medium, High
  - ▶ Set a weighting multiplier for each (0 → 15)
  - ▶ This results in the “unadjusted function-point total”
- ▶ 3. Compute an “influence multiplier” and apply
  - ▶ It ranges from 0.65 to 1.35; is based on 14 factors
- ▶ 4. Results in “function point total”
  - ▶ This can be used in comparative estimates
- ▶ 5. Estimating effort and time
  - ▶ Calculate based on per function point effort

# Example - Online Bookstore Application

---

- ▶ **User Management:** Registration, Login, Profile Update, etc.
- ▶ **Product Management:** Adding new books, Editing book details, etc.
- ▶ **Order Management:** Placing orders, Viewing order history, etc.
- ▶ **Search Functionality:** Searching for books, etc.
- ▶ **Reviews and Ratings:** Adding and viewing reviews, etc.



# Step 1: Identify and Classify Functions

---

- ▶ Inputs - e.g., User Registration, Adding new books
- ▶ Outputs - e.g., Displaying order confirmation
- ▶ Inquiries - e.g., Searching for books
- ▶ Logical Files - e.g., User account details, Book details
- ▶ Interface Files- e.g., Interfaces with a payment gateway



## Step 2: Assign Complexity Weights

---

For example (for inputs):

- ▶ Low complexity: 3 FP
- ▶ Average complexity: 4 FP
- ▶ High complexity: 6 FP



# Function point multipliers

---

|                          | Function Points |                   |                 |
|--------------------------|-----------------|-------------------|-----------------|
| Program Characteristic   | Low Complexity  | Medium Complexity | High Complexity |
| Number of Inputs         | x 3             | x 4               | x 6             |
| Number of Outputs        | x 4             | x 5               | x 7             |
| Inquiries                | x 3             | x 4               | x 6             |
| Logical internal files   | x 7             | x 10              | x 15            |
| External interface files | x 5             | x 7               | x 10            |

# Counting the Number of Function Points

Step 3: Calculate Unadjusted Function Points (UFP)

Step 4: Calculate Adjusted Function Points (AFP)

Influence multiplier is based on 14 general system characteristics like data communications, distributed functions, performance requirements, etc.

Each characteristic is rated on a scale from 0 (no influence) to 5 (strong influence), and the total is summed up

|                                 | Function Points   |                    |                    |
|---------------------------------|-------------------|--------------------|--------------------|
| Program Characteristic          | Low Complexity    | Medium Complexity  | High Complexity    |
| Number of Inputs                | $5 \times 3 = 15$ | $2 \times 4 = 8$   | $3 \times 6 = 18$  |
| Number of Outputs               | $6 \times 4 = 24$ | $6 \times 5 = 30$  | $0 \times 7 = 0$   |
| Inquiries                       | $0 \times 3 = 0$  | $2 \times 4 = 8$   | $4 \times 6 = 24$  |
| Logical internal files          | $5 \times 7 = 35$ | $2 \times 10 = 20$ | $3 \times 15 = 45$ |
| External interface files        | $8 \times 5 = 40$ | $0 \times 7 = 0$   | $2 \times 10 = 20$ |
| Unadjusted function-point total | 287               |                    |                    |
| Influence multiplier            | 1.20              |                    |                    |
| Adjusted function-point total   | 344               |                    |                    |



# Over and Under Estimation

---

## ▶ Over estimation issues

- ▶ The project will not be funded
  - ▶ Conservative estimates guaranteeing 100% success may mean funding probability of zero.
- ▶ Danger of feature and scope creep
- ▶ Be aware of “double-padding”: team member + manager

## ▶ Under estimation issues

- ▶ Quality issues (short changing key phases like testing)
- ▶ Inability to meet deadlines
- ▶ Morale and other team motivation issues

# Wideband Delphi

---

- ▶ Group consensus approach
- ▶ Present experts with a problem and response form
- ▶ Conduct group discussion, collect anonymous opinions, then feedback
- ▶ Conduct another discussion & iterate until consensus
- ▶ Advantages
  - ▶ Easy, inexpensive, utilizes expertise of several people
  - ▶ Does not require historical data
- ▶ Disadvantages
  - ▶ Difficult to repeat
  - ▶ May fail to reach consensus, reach wrong one, or all may have same bias

# PROBLEM STATEMENT

---

The system is an on-line version of the popular Monopoly board game. The game provides many of the features found in the board version of the game. Unless otherwise specified this game follows the standard rules of the board game.



# Scheduling & Tracking

# How To Schedule

---

- ▶ 1. Identify “what” needs to be done
  - ▶ Work Breakdown Structure (WBS)
- ▶ 2. Identify “how much” (the size)
  - ▶ Size estimation techniques
- ▶ 3. Identify the dependency between tasks
  - ▶ Dependency graph, network diagram
- ▶ 4. Estimate total duration of the work to be done
  - ▶ The actual schedule



# Partitioning Your Project

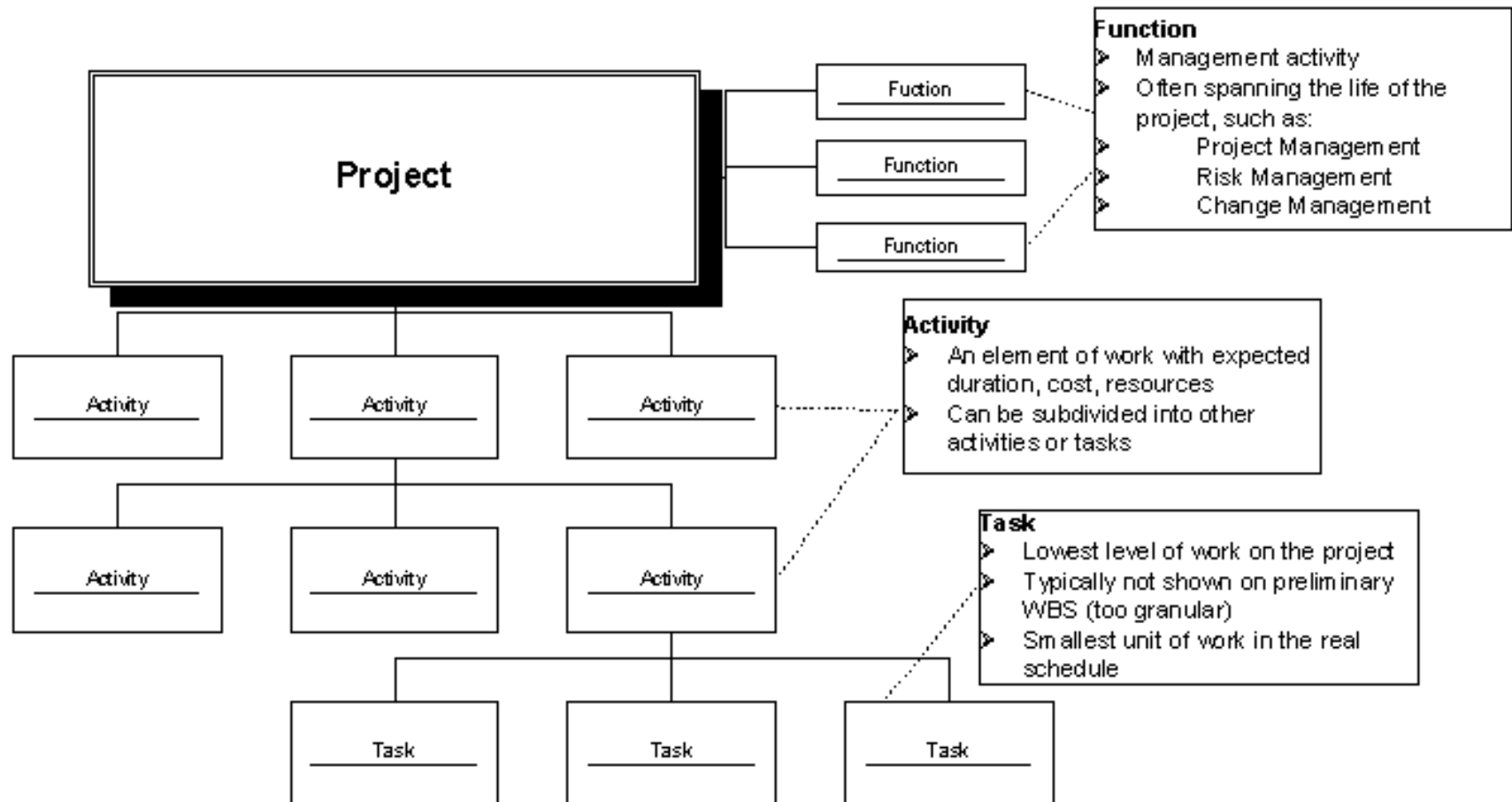
---

- ▶ You need to decompose your project into manageable chunks
- ▶ ALL projects need this step
- ▶ Divide & Conquer
- ▶ Two main causes of project failure
  - ▶ Forgetting something critical
  - ▶ Ballpark estimates become targets
- ▶ How does partitioning help this?



# Project Elements

## ► A Project: functions, activities, tasks



# Work Break Down Structure (WBS)

---

- ***Work Break Down Structure*** – a check list of the work that must be accomplished to meet the project objectives.
- The WBS lists the major project outputs and those departments or individuals primarily responsible for their completion.





# 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

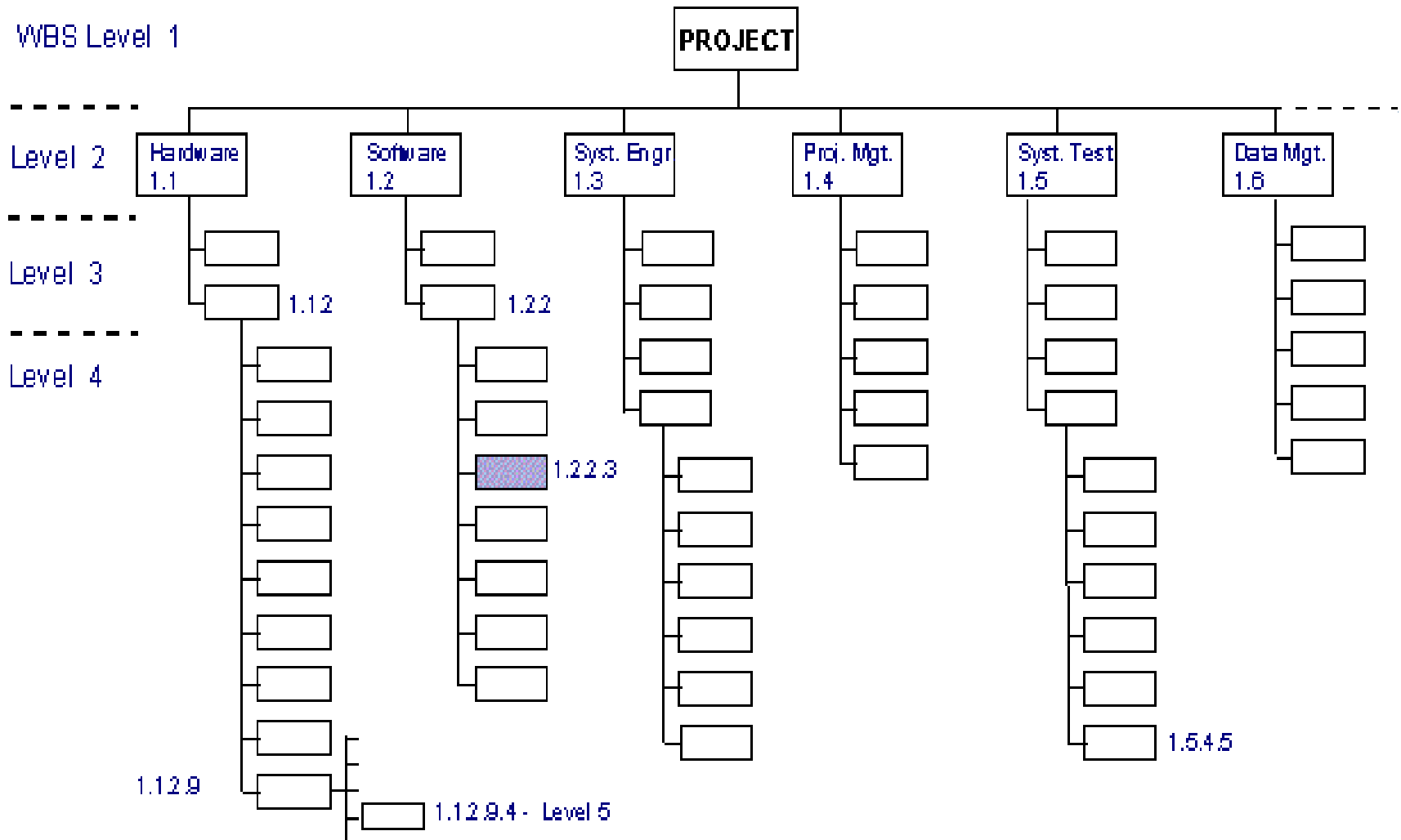
4.2.5 Transaction Processing

4.3 Graphics and Interface

4.4 Content Creation

5.0 Testing and Production





From: [http://www.hyperthot.com/pm\\_wbs.htm](http://www.hyperthot.com/pm_wbs.htm)

# WBS Types

---

- ▶ Process WBS

- ▶ a.k.a Activity-oriented
- ▶ Ex: Requirements, Analysis, Design, Testing
- ▶ Typically used by PM

- ▶ Product WBS

- ▶ a.k.a. Entity-oriented
- ▶ Ex: Financial engine, Interface system, DB
- ▶ Typically used by engineering manager

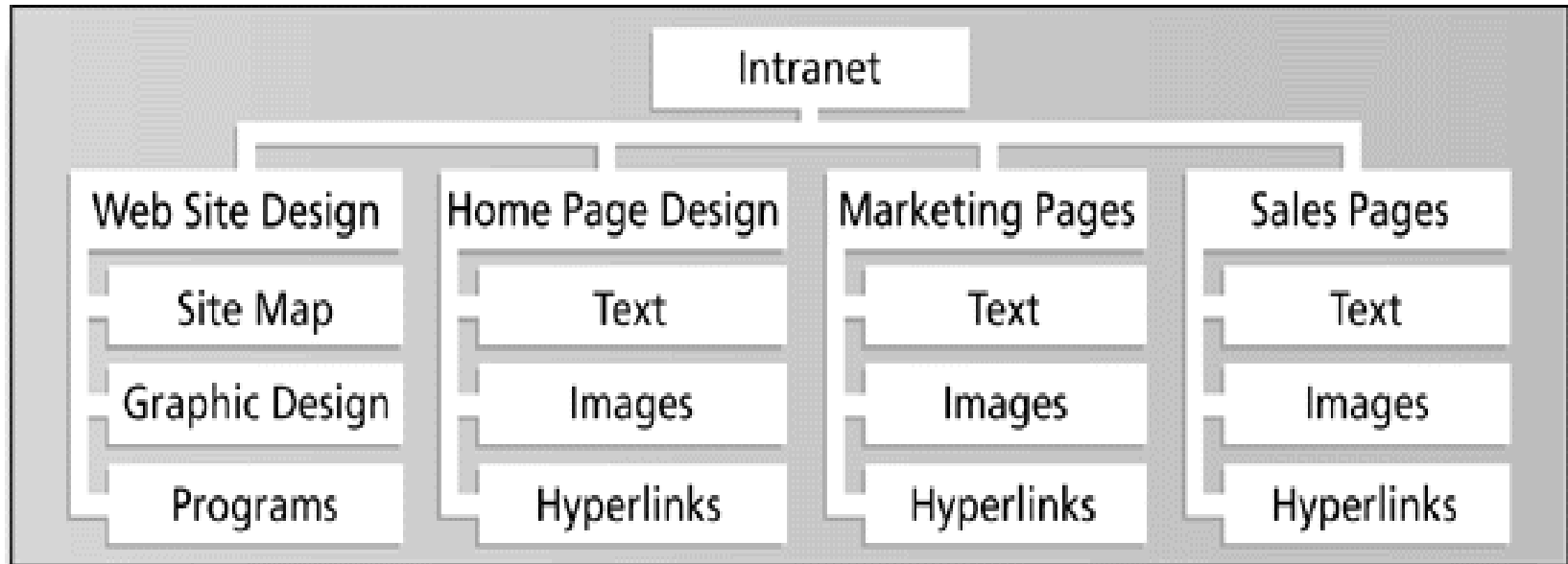
- ▶ Hybrid WBS: both above

- ▶ This is not unusual
- ▶ Ex: Lifecycle phases at high level with component or feature-specifics within phases
- ▶ Rationale: processes produce products



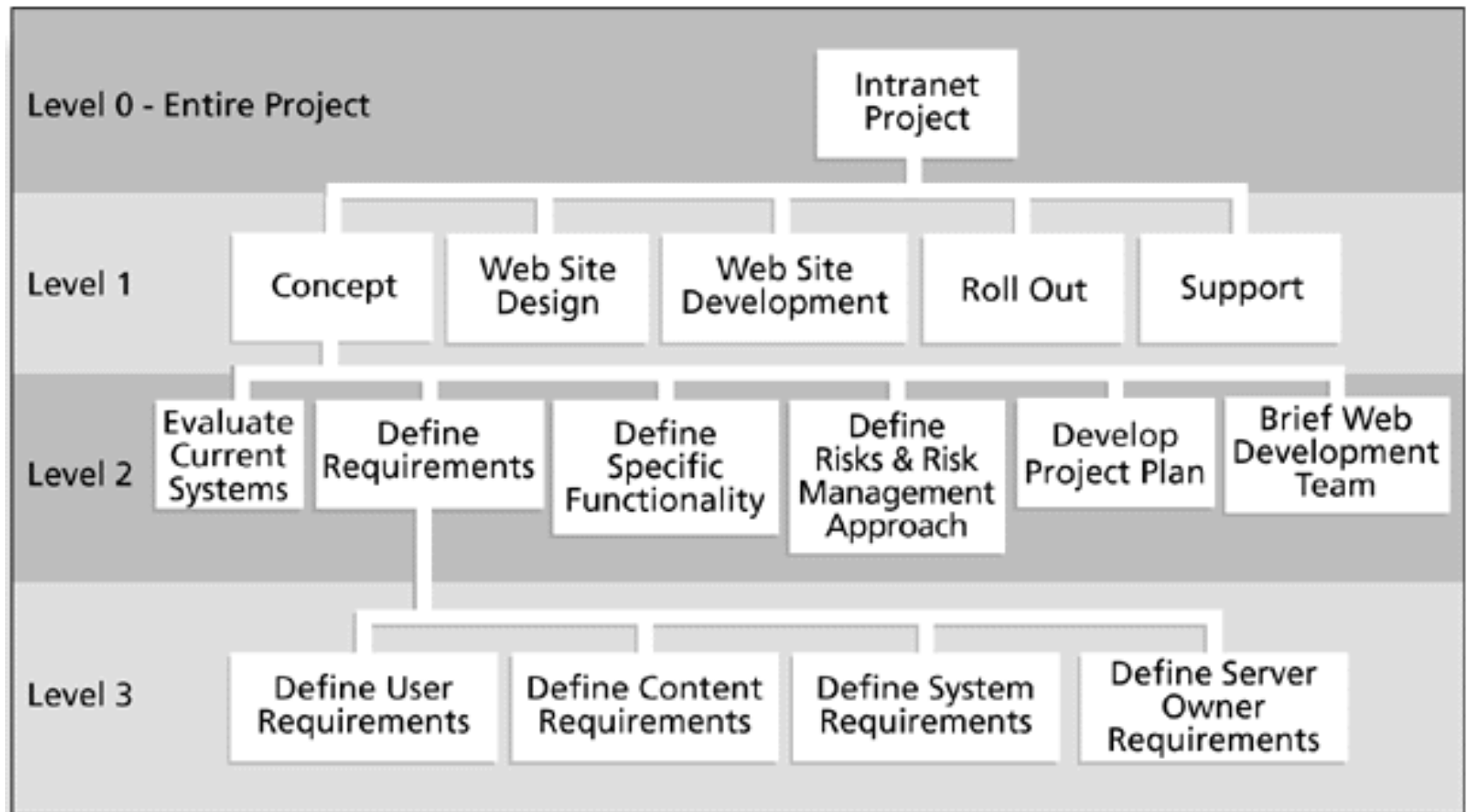
# Product WBS

---



# Process WBS

---



# WBS

---

- ▶ List of Activities, not Things
- ▶ List of items can come from many sources
  - ▶ SOW, Proposal, brainstorming, stakeholders, team
- ▶ Describe activities using “bullet language”
  - ▶ Meaningful but terse labels
- ▶ All WBS paths do not have to go to the same level
- ▶ Do not plan more detail than you can manage



# Work Packages (Tasks)

---

- ▶ Generic term for discrete **tasks** with definable end results
- ▶ The “one-to-two” rule
  - ▶ Often at: 1 or 2 persons for 1 or 2 weeks
- ▶ Basis for monitoring and reporting progress
  - ▶ Can be tied to budget items (charge numbers)
  - ▶ Resources (personnel) assigned
- ▶ Ideally shorter rather than longer
  - ▶ Not so small as to micro-manage



# WBS Techniques

---

- ▶ Top-Down
- ▶ Bottom-Up
- ▶ Analogy
- ▶ Rolling Wave
  - ▶ 1<sup>st</sup> pass: go 1-3 levels deep
  - ▶ Gather more requirements or data
  - ▶ Add more detail later
- ▶ Post-its on a wall





# WBS Techniques

---

- ▶ Analogy

- ▶ Base WBS upon that of a “similar” project
- ▶ Use a template
- ▶ Analogy also can be estimation basis
- ▶ Pros
  - ▶ Based on past actual experience
- ▶ Cons
  - ▶ Needs comparable project



# Sequence the Work Activities

---

- ✦ Milestone Chart
- ✦ Gantt chart
- ✦ Network Techniques
  - CPM (Critical Path Method)
  - PERT (Program Evaluation and Review Technique)

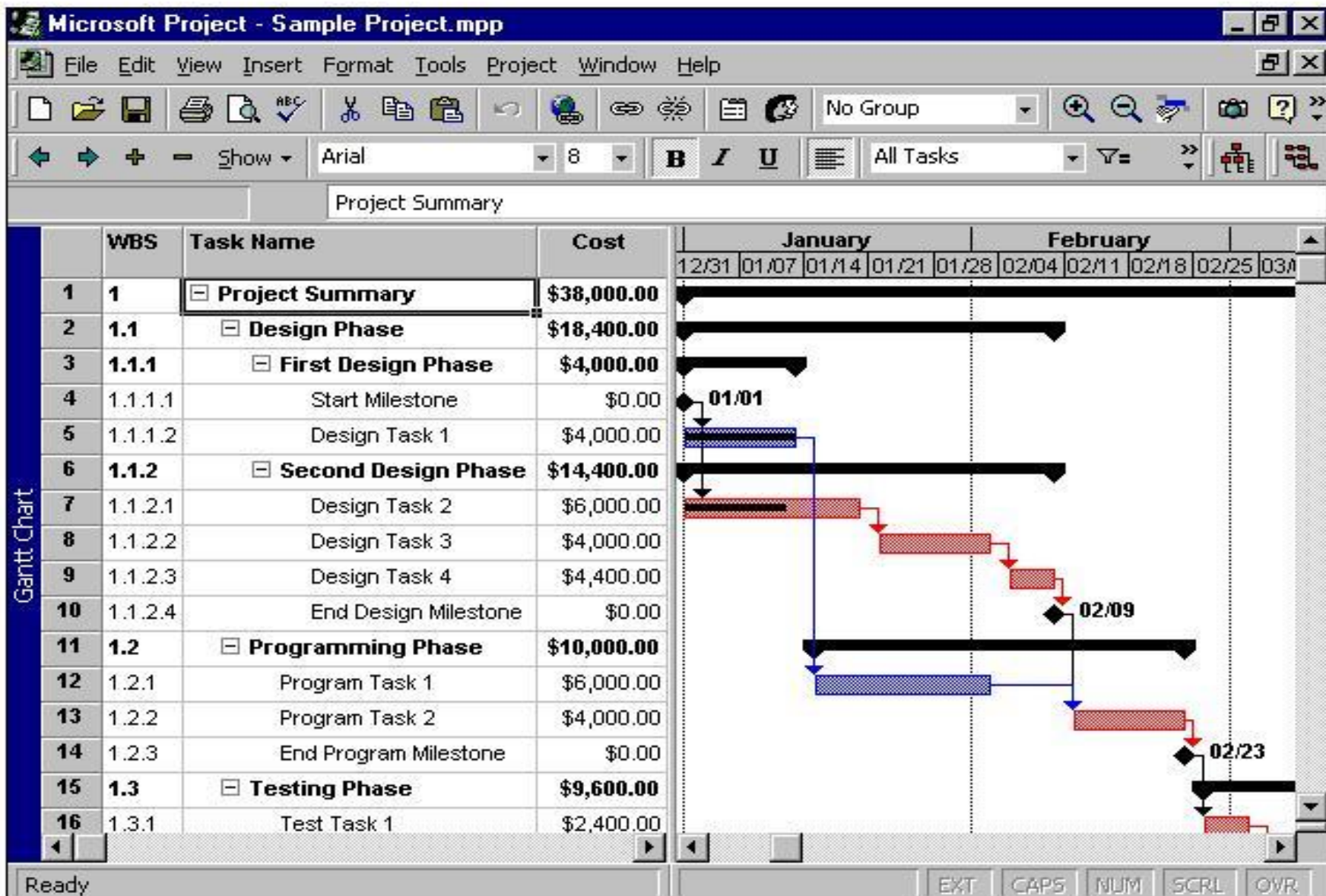


# Gantt Chart

---

- Gantt chart is a means of displaying simple activities or events plotted against time or dollars
- Most commonly used for exhibiting program progress or for defining specific work required to reach an objective
- Gantt charts may include listing of activities, activity duration, scheduled dates, and progress-to-date





# Gantt Chart

---

- Advantages:
  - Easy to understand
  - Easy to change
- Disadvantages:
  - only a vague description of the project
  - does not show interdependency of activities
  - cannot show results of an early or late start of an activity

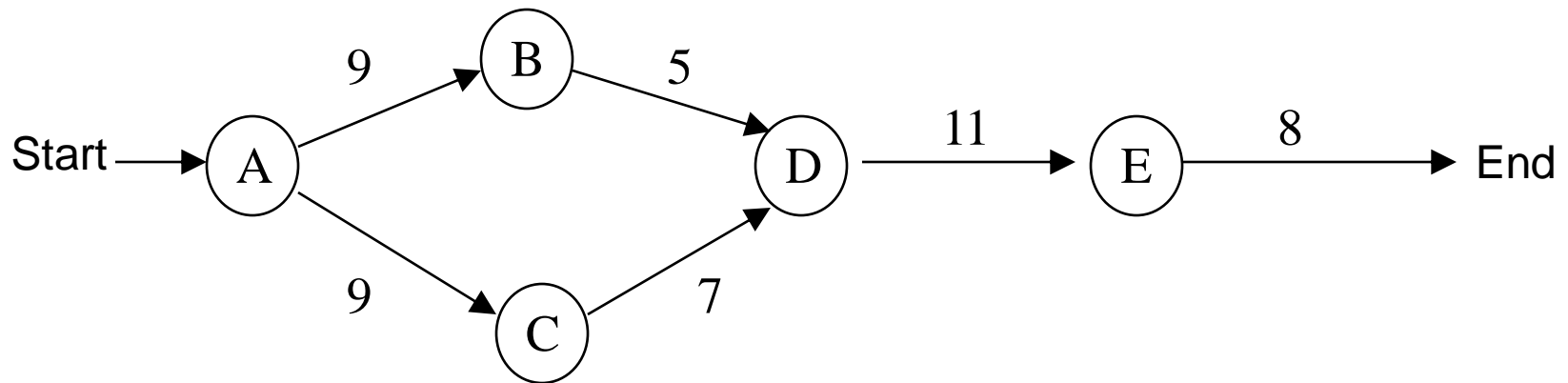
# Network Techniques

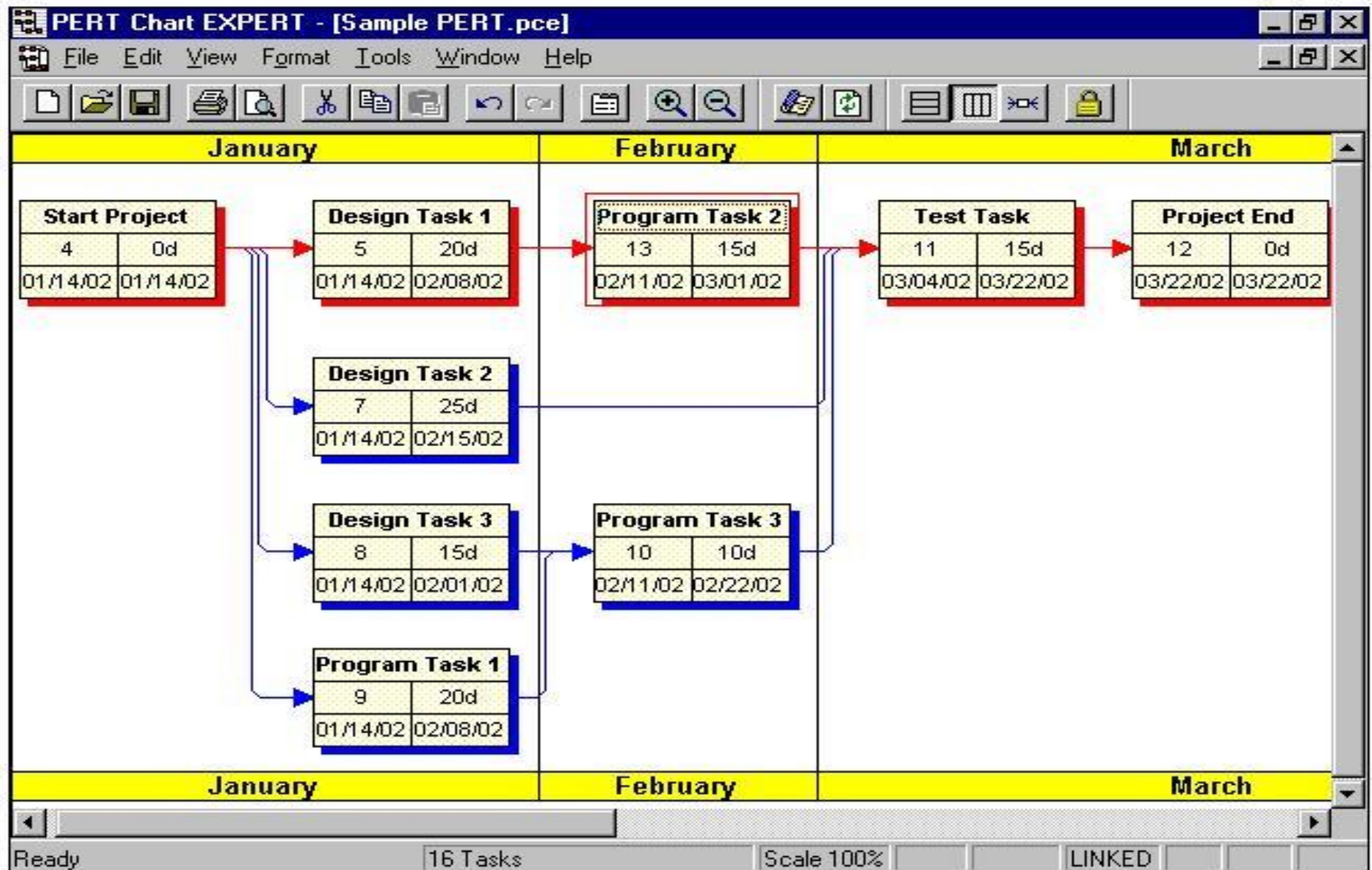
---

- A *precedence network* diagram is a graphic model portraying the sequential relationship between key events in a project.
- Initial development of the network requires that the project be defined and thought out.
- The network diagram clearly and precisely communicates the plan of action to the project team and the client.



| Task                               | Duration | Dependencies |
|------------------------------------|----------|--------------|
| A - Architecture & design strategy | 9        | start        |
| B - Decide on number of releases   | 5        | A            |
| C - Develop acceptance test plan   | 7        | A            |
| D - Develop customer support plan  | 11       | B,C          |
| E - Final sizing & costing         | 8        | D            |







# Critical Path Method (CPM)

---

CPM tries to answer the following questions:

1. What is the duration of the project?
2. By how much (if at all) will the project be delayed if any one of the activities takes  $N$  days longer?
3. How long can certain activities be postponed without increasing the total project duration?



# Critical Path

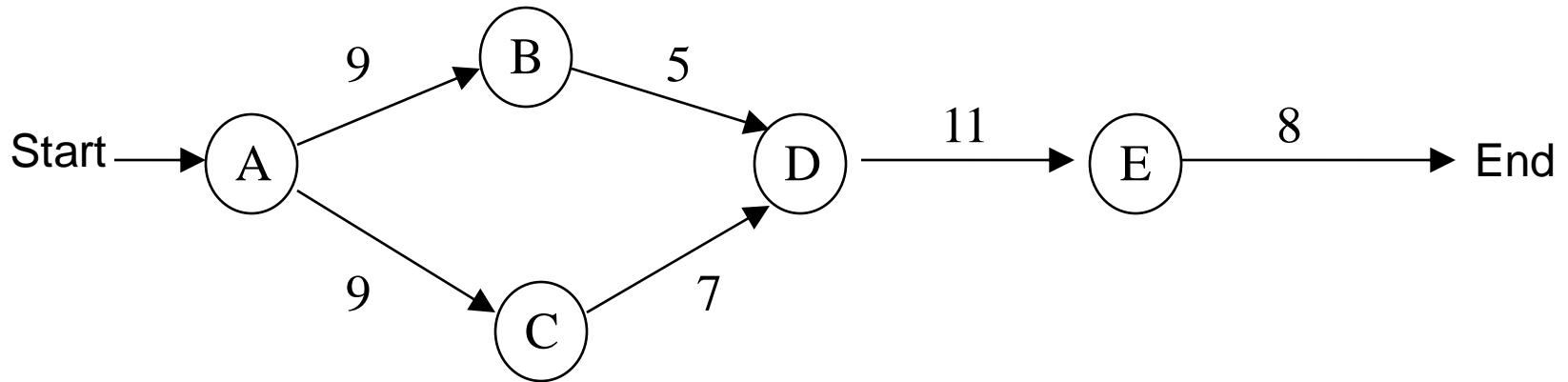
---

- Sequence of activities that have to be executed one after another
- Duration times of these activities will determine the overall project time, because there is no slack/float time for these activities
- If any of the activities on the critical path takes longer than projected, the entire project will be delayed by that same amount
- Critical path = Longest path in the precedence network (generally, the longest in time)



# Critical Path

---



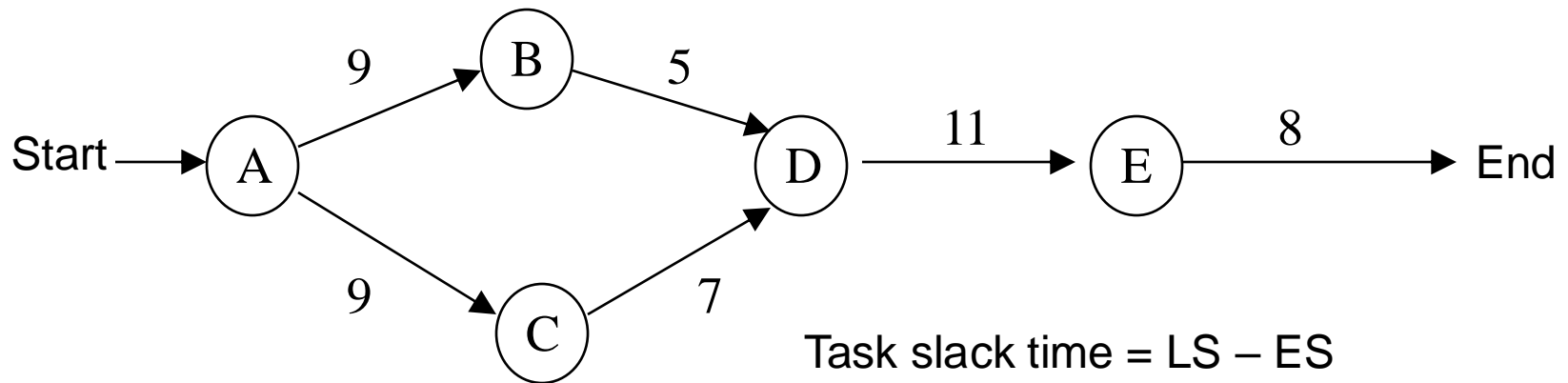
Critical Path = A – C – D – E (35 time units)

Critical Tasks = A,C,D,E

Non-Critical Path = A-B-D-E



| Task | Duration | Depend | Earliest Start | Earliest Finish | Latest Start | Latest Finish |
|------|----------|--------|----------------|-----------------|--------------|---------------|
| A    | 9        | none   | 0              | 9               | 0            | 9             |
| B    | 5        | A      | 9              | 14              | 11           | 16            |
| C    | 7        | A      | 9              | 16              | 9            | 16            |
| D    | 11       | B,C    | 16             | 27              | 16           | 27            |
| E    | 8        | D      | 27             | 35              | 27           | 35            |



**Slack time** – maximum allowable delay for a non-critical activity.

$$\text{Task slack time} = \text{LS} - \text{ES}$$

- or -

$$\text{Task slack time} = \text{LF} - \text{EF}$$

Task B has 2 time units of **slack time**