SOFTWARE PROJECT PLANNING AND MANAGEMENT

Prof. Sudip Misra
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur
http://cse.iitkgp.ac.in/~smisra/

Necessity of Project Management

- Essential for developing a project.
- Essential for managing resources.
- Essential to maintain the cost, list of deliverables, and dead lines.
- Essential for defining the strategies.

Result of Bad Project Management

- Lack of skillful resources.
- Substandard quality.
- Crossing the deadline.
- Budget overshoot.
- Chaotic management.
- High risk.

Goal of Project Management

"The main goal of software project management is to enable a group of developers to work effectively towards the successful completion of a project ."

Ref: Fundamentals of Software Engineering, Rajib Mall

Complexities of Project Management

- Invisibility.
- Changeability.
- Complexity.
- Uniqueness.
- Exactness of the solution.

Key players of project management

- Manager.
- Team leader.

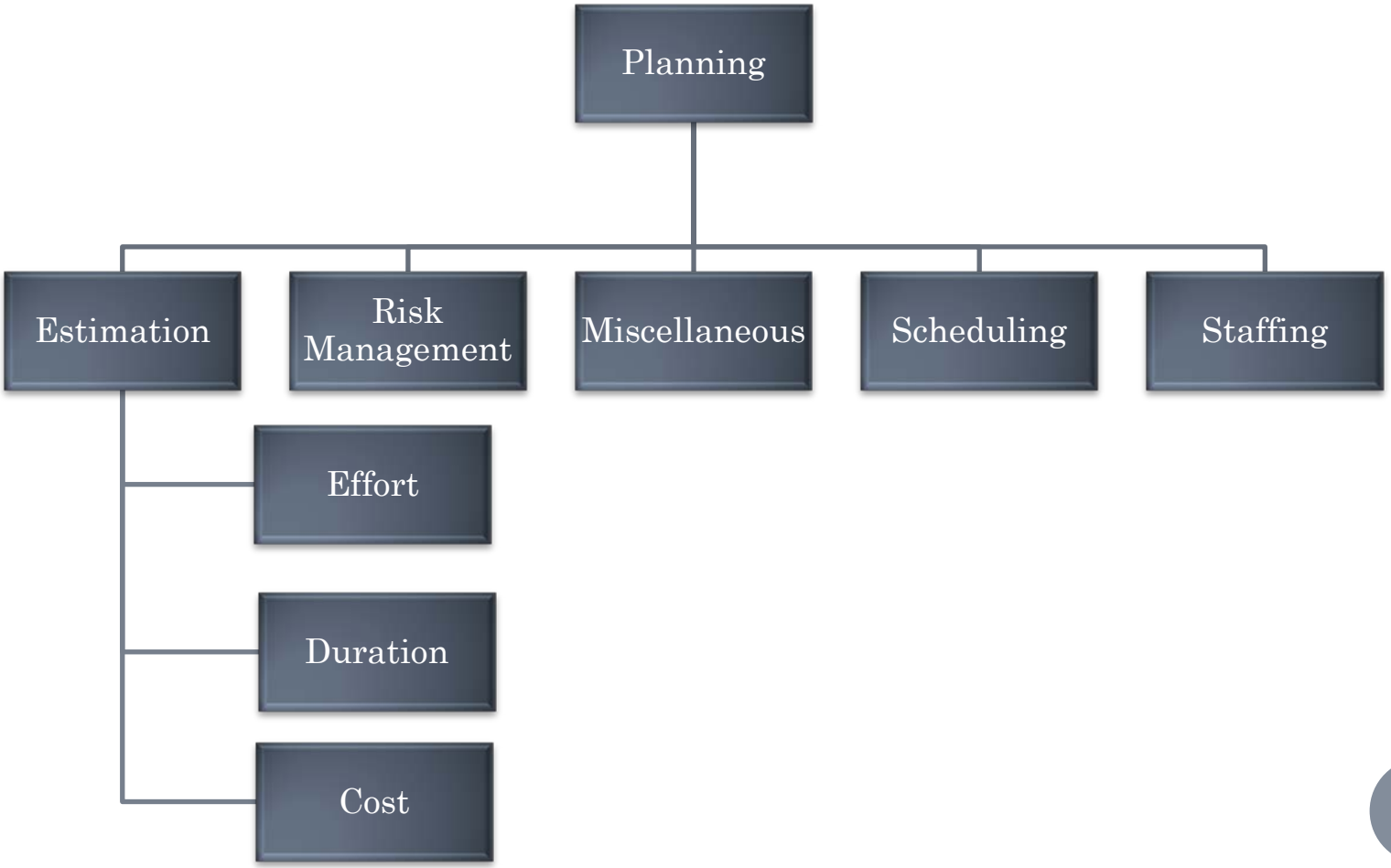
Roles of manager

- Job responsibilities.
- Necessary skill for maintaining project.

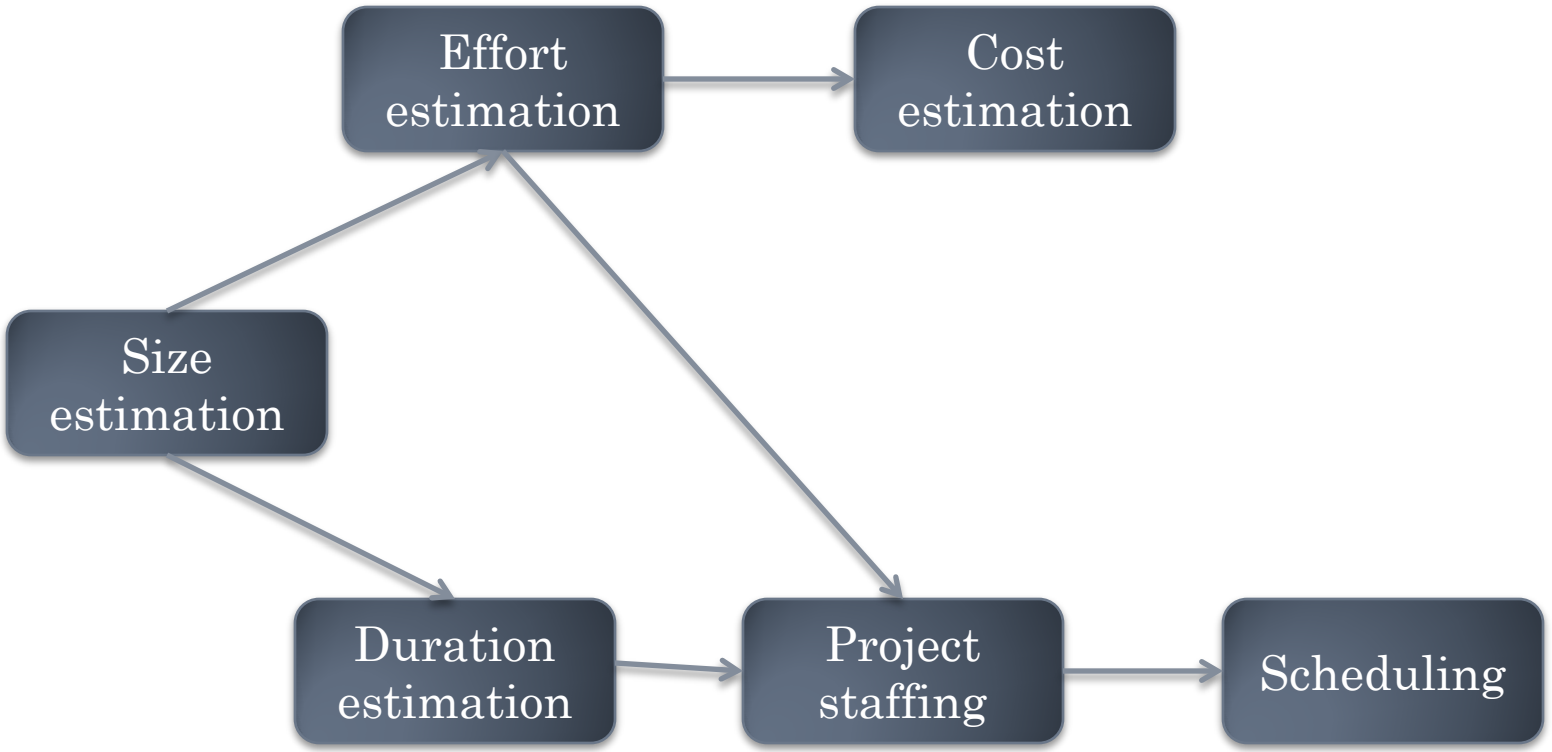
Roles of manager: Job responsibilities

- Project planning.
- Project monitoring.
- Control.

PROJECT PLANNING



Steps of planning activities



Sliding Window Planning

- "In the sliding window planning technique, starting with an initial plan, the project is planned more accurately over a number of stages."

Ref: Fundamentals of Software Engineering, Rajib Mall

SPMP Document

Organisation of the software project management plan (SPMP) document

1. **Introduction**
 (a) Objectives
 (b) Major Functions
 (c) Performance Issues
 (d) Management and Technical Constraints
2. **Project estimates**
 (a) Historical Data Used
 (b) Estimation Techniques Used
 (c) Effort, Resource, Cost, and Project Duration Estimates
3. **Schedule**
 (a) Work Breakdown Structure
 (b) Task Network Representation
 (c) Gantt Chart Representation
 (d) PERT Chart Representation
4. **Project resources**
 (a) People
 (b) Hardware and Software
 (c) Special Resources
5. **Staff organisation**
 (a) Team Structure
 (b) Management Reporting
6. **Risk management plan**
 (a) Risk Analysis
 (b) Risk Identification
 (c) Risk Estimation
 (d) Risk Abatement Procedures
7. **Project tracking and control plan**
 (a) Metrics to be tracked
 (b) Tracking plan
 (c) Control plan
8. **Miscellaneous plans**
(a) Process Tailoring
(b) Quality Assurance Plan
(c) Configuration Management Plan
(d) Validation and Verification
(e) System Testing Plan
(f) Delivery, Installation, and Maintenance Plan

METRICS FOR PROJECT SIZE ESTIMATION

- Lines Of Code (LOC).
- Function Point (FP).

Metrics of Lines Of Code (LOC)

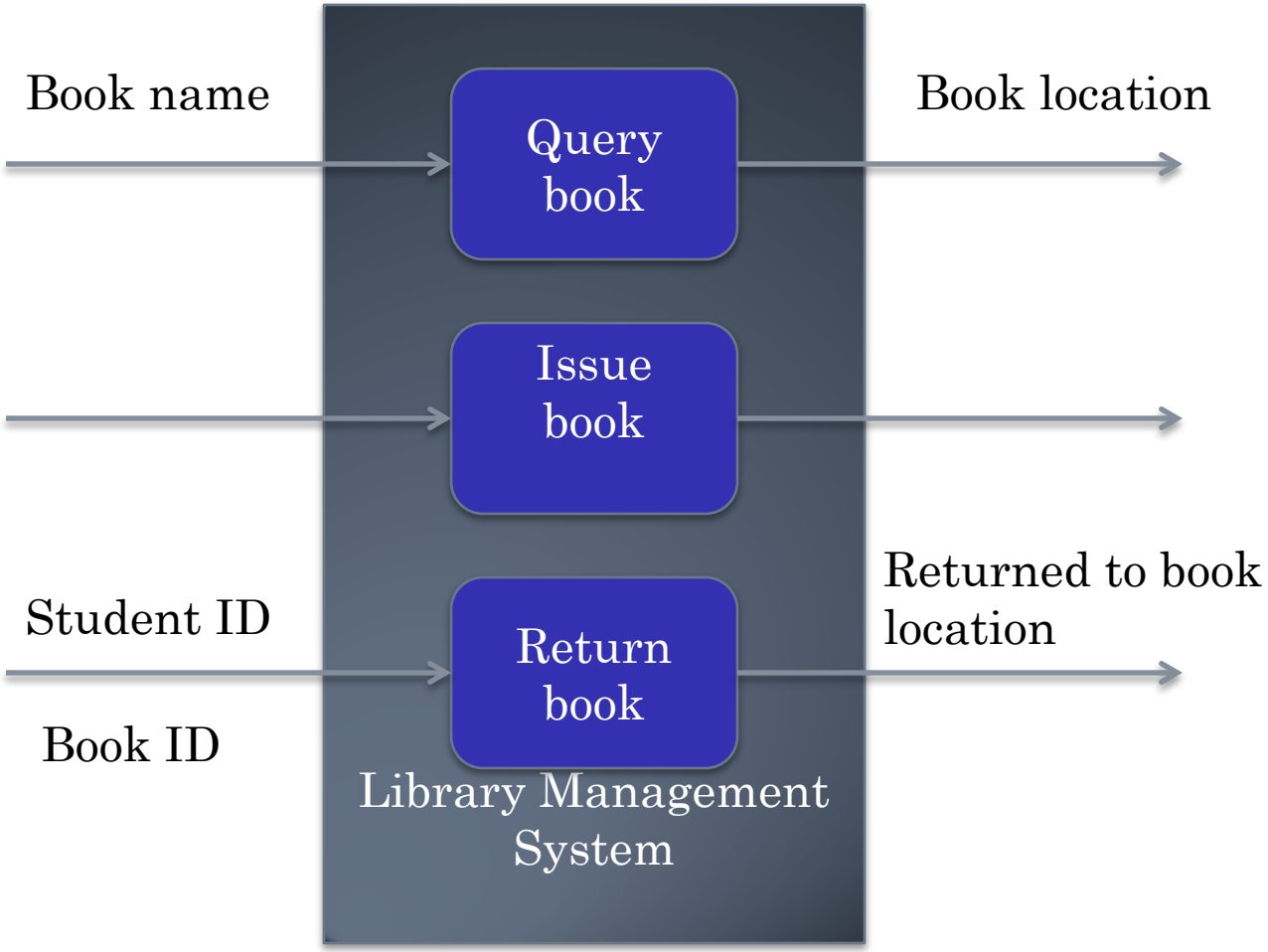
- Choice of specific instructions.
- Quality and efficiency.
- Code reuse.
- Lexical complexity.

FUNCTION POINT (FP) METRIC

"Conceptually, the function point metric is based on the idea that a software product supporting many features would certainly be of larger size than a product with less number of features."

Ref: Fundamentals of Software Engineering, Rajib Mall

Example of FP



PROJECT ESTIMATION TECHNIQUES

- Empirical.
- Heuristic.
- Analytical.

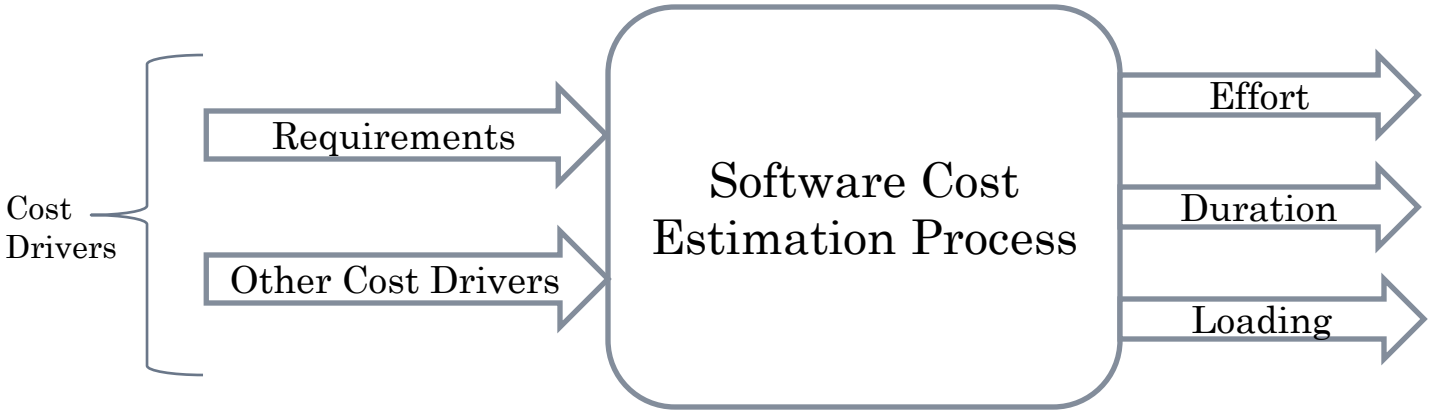
Empirical Estimation

- Expert judgement.
- Delphi cost estimation.

Heuristic estimation : Cost Estimation

- Cost of a projects involves
  - The cost of tools and hardware required for the development.
  - The cost of human resources required for the development.
- The software cost estimation provides
  - the vital link between the general concepts and techniques of **economic analysis** and the particular world of **software engineering.**
  - Software cost estimation techniques also provides an essential part of the foundation for **good software management**.

Cost Estimation Process



Classical view of software cost estimation process

COCOMO MODEL

- The **Constructive Cost Model** (COCOMO) is most widely used software estimation model.
- COCOMO represents a comprehensive empirical model for software cost estimation.
- The hierarchy of COCOMO models takes the following form.
  - *Model 1*. The Basic COCOMO model
  - *Model 2*. The Intermediate COCOMO model
  - *Model 3*. The Advanced COCOMO model

COCOMO MODEL

- Basic COCOMO model is single-valued model, that computes software development effort (and cost) as a function of program size or lines of code (LOC).
- Intermediate COCOMO model computes software development effort as a function of *program size* and a set of "*cost drivers*" (subjective assessments of product, hardware, personnel and project attributes).
- Advanced COCOMO model similar to intermediate model, with an assessment of the cost driver's impact on each step (analysis, design, etc.).

COCOMO MODEL

- Defined for three classes of software projects.
  - *organic mode*– relatively small, simple software projects in which small teams with good application experience work to a set of less than rigid requirements (e.g., a thermal analysis program developed for a heat transfer group).
  - *semi-detached mode*– an intermediate (in size and complexity) software project in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements (e.g., a transaction processing system with fixed requirements for terminal hardware and data base software).
  - *embedded mode*– a software project that must be developed within a set of tight hardware, software and operational constraints (e.g., flight control software for aircraft).

Basic Cocomo Model

- The Basic COCOMO equations take the form:
  - $E = a_b(KLOC)^{b_d}$
  - $D = c_b E^{d_b}$
    - where E is the effort applied in person-months.
    - KLOC: Program size in kilo line of Code
    - D is the duration of the project.
    - $a_b, b_d, c_b, and\ d_b$ are constants.

Software Project	a_b	b_d	c_b	d_b
organic	2.4	1.05	2.5	0.38
semi-detached	3.0	1.12	2.5	0.35
embedded	3.6	1.20	2.5	0.32

Intermediate Cocomo Model

- Extends the basic model by consider by other *"cost driving attributes"*.
- Cost driving attributes falls in four categories.
  - Product attributes (3)
  - Hardware attributes (4)
  - Personnel attributes (5)
  - Project attributes (5)
- Each of the 15 attributes is rated on a 6 point scale that ranges from "very low" to "extra high".
- Based on the rating, an effort multiplier is determined from tables published by Boehm [BOE81]
- the product of all effort multipliers results is an *effort adjustment factor* (EAF) [0.9 to 1.4].

Intermediate Cocomo Model

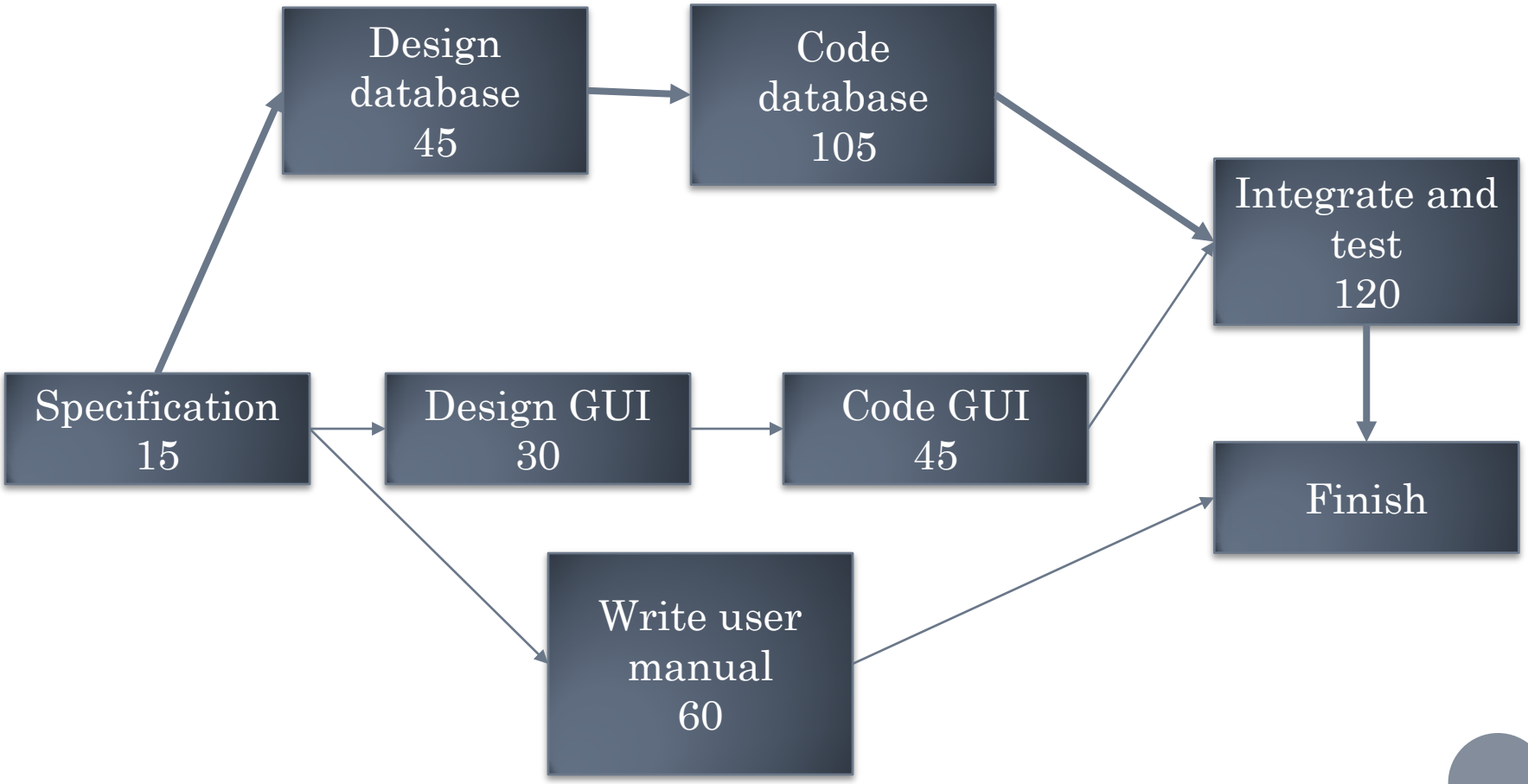
- The intermediate COCOMO model takes the form.
  - $E = a_i(KLOC)^{b_i}(EAF)$
    - Where, EAF is *effort adjustment factor* [0.9 to 1.4].

Software Project	a_i	b_i
organic	3.2	1.05
semi-detached	3.0	1.12
embedded	2.8	1.20

Activity Networks

- It shows –
  - different activities making up a project.
  - estimated duration.
  - interdependencies of different activities.

Activity Network Representation of Management Information System (MIS)



Activity Network Representation of Management Information System (MIS)

Project parameters of MIS project from Activity Network representation

Task No	Task	Duration (hours)	Dependent on task
T1	Specification	15	-
T2	Design database	45	T1
T3	Design GUI	30	T1
T4	Code database	105	T2
T5	Code GUI part	45	T3
T6	Integrate and test	120	T4 and T5
T7	Write user manual	60	T1

Critical Path Method (CPM)

"A critical task is one with a zero slack time. A path from the start node to the finish node containing only critical tasks is called a critical path."

Ref: Fundamentals of Software Engineering, Rajib Mall

Critical Path Method (CPM)

This algorithm determines –

- Critical paths.
- Slack times.

CPM QUANTITIES

- **Minimum time (MT)**
- **Earliest start (ES)**
- **Latest start time (LST)**
- **Earliest finish time (EF)**
- **Latest finish (LF)**
- **Slack time (ST)**

CRITICAL PATH

- The critical path is identified by finding the tasks where ES = LS.
- These are tasks for which there is no ‘slack’ or spare time for a task in the project.

Slack time

- Slack is calculated by LS - ES.

PERT CHART

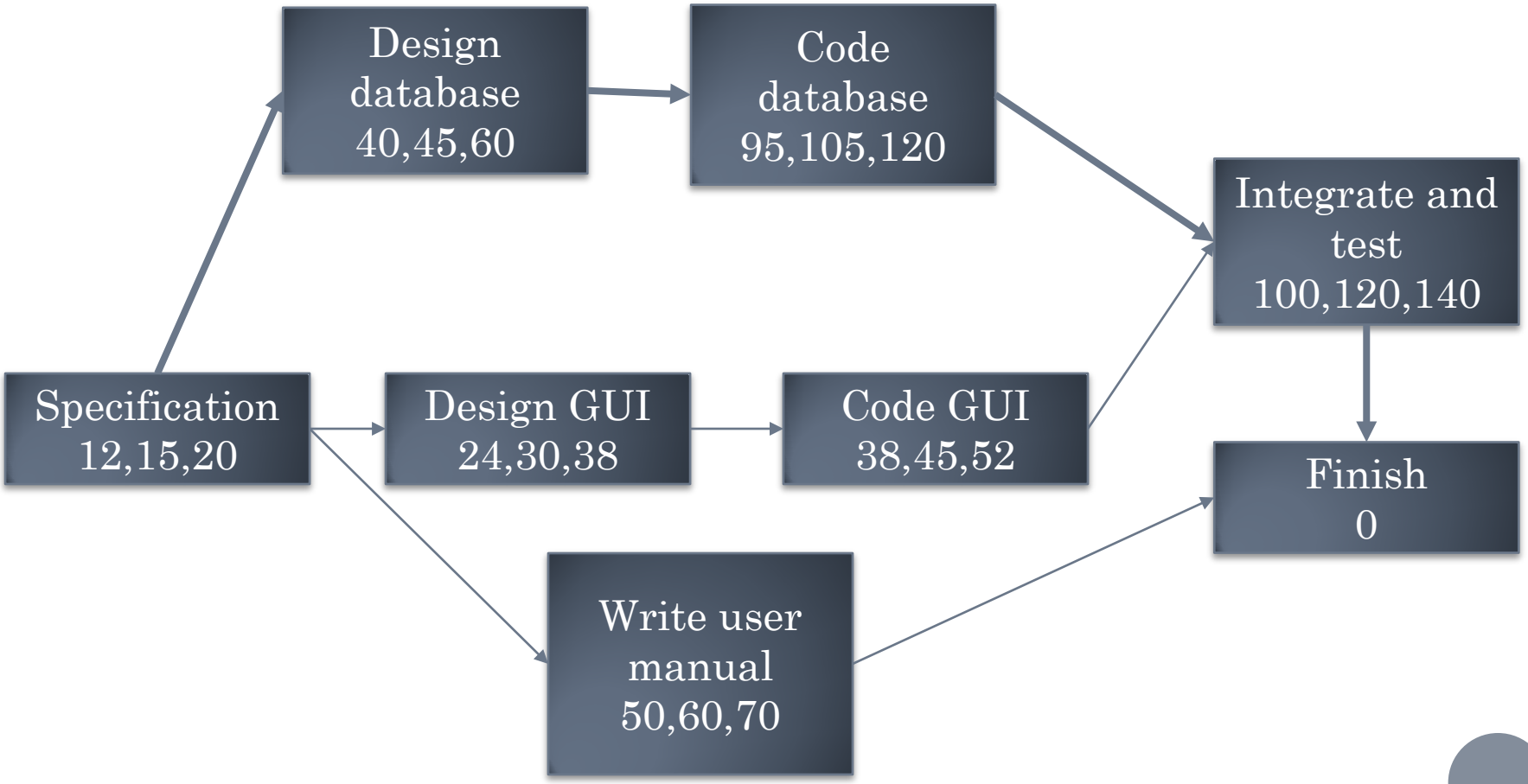
Project Evaluation and Review Technique (PERT) is a graphical representation of a project timeline in a sophisticated way.

PERT CHART

Each task is annotated with three estimates for PERT chart. These are –

- Optimistic (O).
- Most likely estimate (M).
- Worst case (W).

PERT CHART REPRESENTATION OF THE MIS PROBLEM.



Gantt Charts

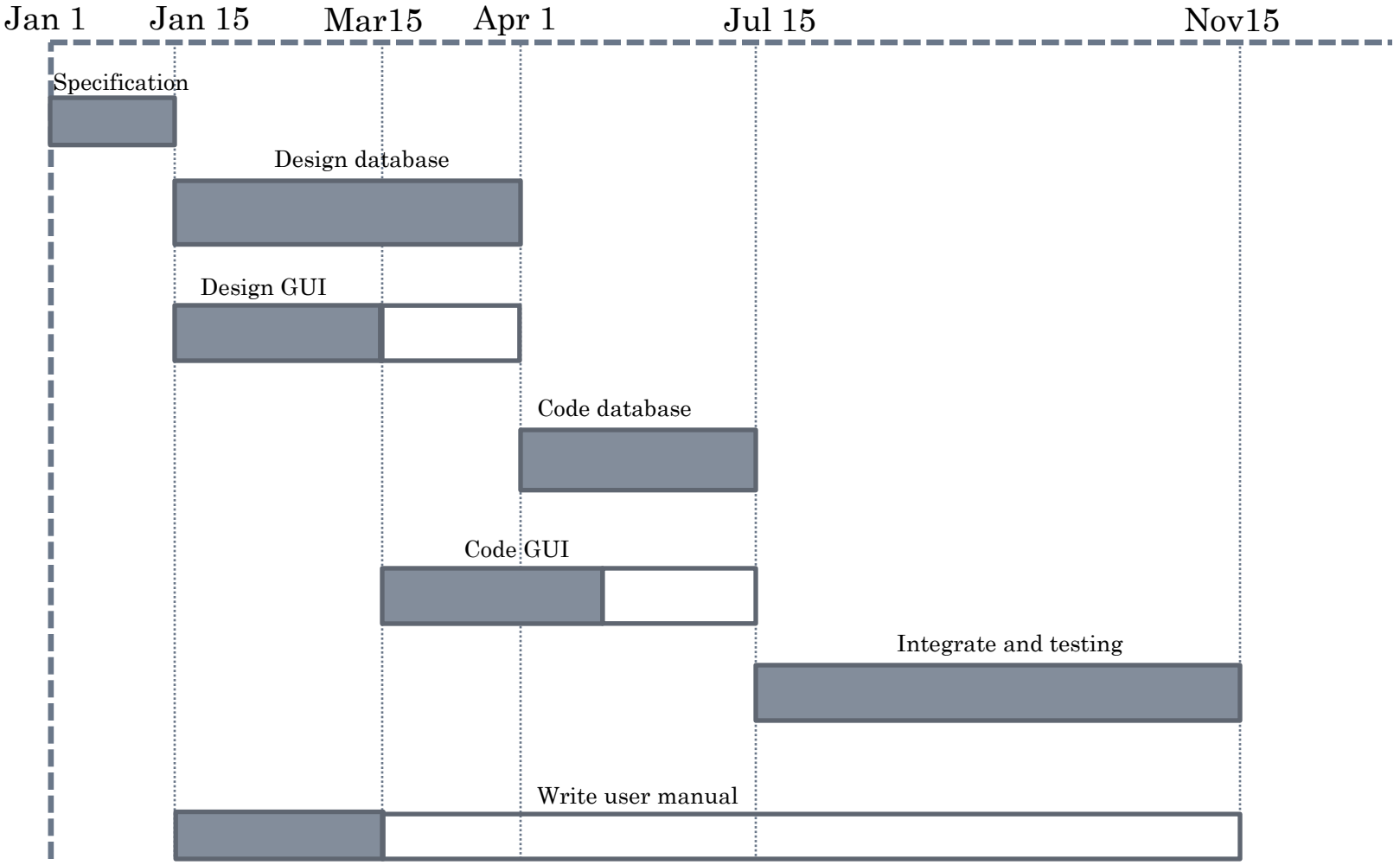
"A Gantt chart is a special type of bar chart where each bar represents an activity. The bars are drawn along a time line. The length of each bar is proportional to the duration of time planned for the corresponding activity."

Ref: Fundamentals of Software Engineering, Rajib Mall

Gantt Charts

- Named after its developer Henry Gantt.
- Useful for planning and scheduling projects.
- Helps to manage the dependencies between tasks.
- Each task is represented by each bar which is drawn along with the Y axis.
- Shaded part shows the estimated time.
- Unshaded part shows the slack time.

Gantt Charts



Draw a Gantt Chart

Project start date : 01/01/2019			
Task ID	**Task Description**	**Predecessor Task(s)**	**Time (days)**
1	Establish project	-	2
2	Establish customer requirements	1	3
3	Produce software specification documents	2	4
4	Write test plans	3	1
5	Write code	3	2
6	Developer testing	5	2
7	System testing	4,6	4
8	Write customer documentation	3	3

THANK YOU