Parul® University
Vadodara, Gujarat

NAAC GRADE A++

CSE- AI & DS

Unit-2

Software Project Management

Prof. Deepak Palavalasa
Assistant Professor
AI&DS Department

Contents- Software Testing

1. Software Project Management:
2. Management Spectrum, People, Product, Process-Project
3. W5HH Principle, Importance of Team Management

4. Planning a Software Project :
5. Scope and Feasibility
6. Effort Estimation, Schedule and staffing, Quality Planning
7. Risk management- identification,assessment, control, project monitoring plan, Detailed Scheduling

Management Spectrum

Software Project Management revolves around **4 main elements**:

1. People
2. Product
3. Process
4. Project

Management Spectrum

People
People are the most important part of any software project.
Their skills, communication, teamwork, and responsibility for project success.
People include developers, testers, analysts, designers, project managers, etc.

Product
The product refers to the *actual software* being developed and its features, functions, constraints, and requirements.

Real-Time Example: Product = **Online Shopping Website**
Features include:
- User Login
- Product Search
- Add to Cart

Management Spectrum

Process

Process defines the steps, methods, and models used for development (Agile, Waterfall, Spiral). It ensures disciplined and structured work.

Project

A project includes planning, scheduling, budgeting, risk handling, monitoring, and delivering the product on time.

Real-Time Example:

Developing a **Banking App** with phases:

- Requirement Analysis – 2 weeks
- UI Design – 1 week
- Development – 8 weeks
- Testing – 3 weeks
- Deployment – 1 week

W5HH Principle

A project planning technique to answer important questions:

Why is the project being done?
What will be done?
When will it be completed?
Who will do it?
Where will development happen?
How will it be done?
How much will it cost?

W5HH Principle

Real-time Example:
Project: Building an **Online Learning Platform**

Why: To provide video classes to students
What: Video lessons, tests, tracking progress
When: 6 months
Who: 1 PM, 6 developers, 2 testers
Where: Hybrid (office + remote)
How: Using MERN stack + Cloud
How much: ₹25–30 lakhs

W5HH Principle

Project: Building a Banking Mobile App

Why:
To provide customers secure online banking services
Reduce branch visits & improve convenience

What:
Account balance, statements
Money transfer (IMPS/NEFT/RTGS/UPI)
Bill payments
Card management
Notifications & alerts

W5HH Principle

When:
8 months

Who:
1 PM
6 Developers
3 Testers
1 UI/UX designer
1 Cybersecurity specialist

Where:
Hybrid (office for secure tasks + remote coding)

W5HH Principle

How:
Tech: Flutter/React Native, Node.js/Spring Boot, PostgreSQL
Cloud deployment (AWS/Azure)
High-security encryption + MFA

How much:
₹40–50 lakhs

Project Scope and Feasibility

Project scope defines:

What the project will deliver (software features, modules, outputs)
What is NOT included
Overall **boundaries**, constraints, assumptions

Key Components of Scope

1. Business Requirements
2. Functional Scope
3. Non-Functional Scope
4. Technology and Platform
5. Constraints (Time, Budget, Tools, Regulations)
6. Assumptions

Project Scope and Feasibility

Example – Food Delivery App (like Swiggy/Zomato)
In Scope

- User registration/login
- Browse restaurants and menus
- Cart & online payment
- Order tracking
- Admin dashboard

Feasibility Analysis

Project Feasibility means checking whether a project **can be done successfully** with the available **time, money, resources, technology, and skills**.

It helps decide if the project is **realistic**, **practical**, and **worth doing**.

Feasibility Analysis

Example 1: Building a Mobile App

Before creating a mobile app, you check:

- Do we have skilled developers? (Technical feasibility)
- Do we have enough budget? (Economic feasibility)
- Can it be completed in 3 months? (Schedule feasibility)
- Is it allowed by law? (Legal feasibility)

If all answers are YES → **the project is feasible**.
If not → the project may fail.

Effort Estimation

Effort estimation predicts:

1. **How much work** is required
2. **How many people** are needed
3. **How long** the project will take

Popular Estimation Techniques

1. Expert Judgment
Based on senior developers or architects' experience
Example: “This login module will take around 3 days based on previous projects.”

2. Work Breakdown Structure (WBS)
Break project into **small tasks** → estimate each.
Example – E-commerce Website
UI Design – 40 hours
Backend APIs – 120 hours
Database design – 20 hours
Testing – 50 hours

Popular Estimation Techniques

3. Use Case Point (UCP)
Based on complexity of use cases.
Example:
Login (simple), Payment (complex), Order tracking (medium)

4. Function Point Analysis (FPA)

5. COCOMO Model (Constructive Cost Model)

COCOMO Model (Constructive Cost Model)

COCOMO is a **software estimation model developed by Barry Boehm** used to estimate:

1. **Effort** (person-months)
2. **Development time** (schedule)
3. **Staffing**
4. **Project cost**

COCOMO uses the **size of the software** (measured in KLOC = Thousands of Lines of Code) to determine the effort required.

COCOMO Model (Constructive Cost Model)

Types of COCOMO

Type	Description
Basic COCOMO	Estimates effort based only on size (KLOC)
Intermediate COCOMO	Considers size + cost drivers
Detailed COCOMO	Considers size + cost drivers + phase-wise distribution

Basic COCOMO Model Formula

Effort (E)

$$\text{Effort} = a1 * (KLOC)^{a2} \text{ PM}$$

E → **Effort** in Person-Months
KLOC → Thousands of Lines of Code
a1, a2 → constants based on project type

Development Time (D)

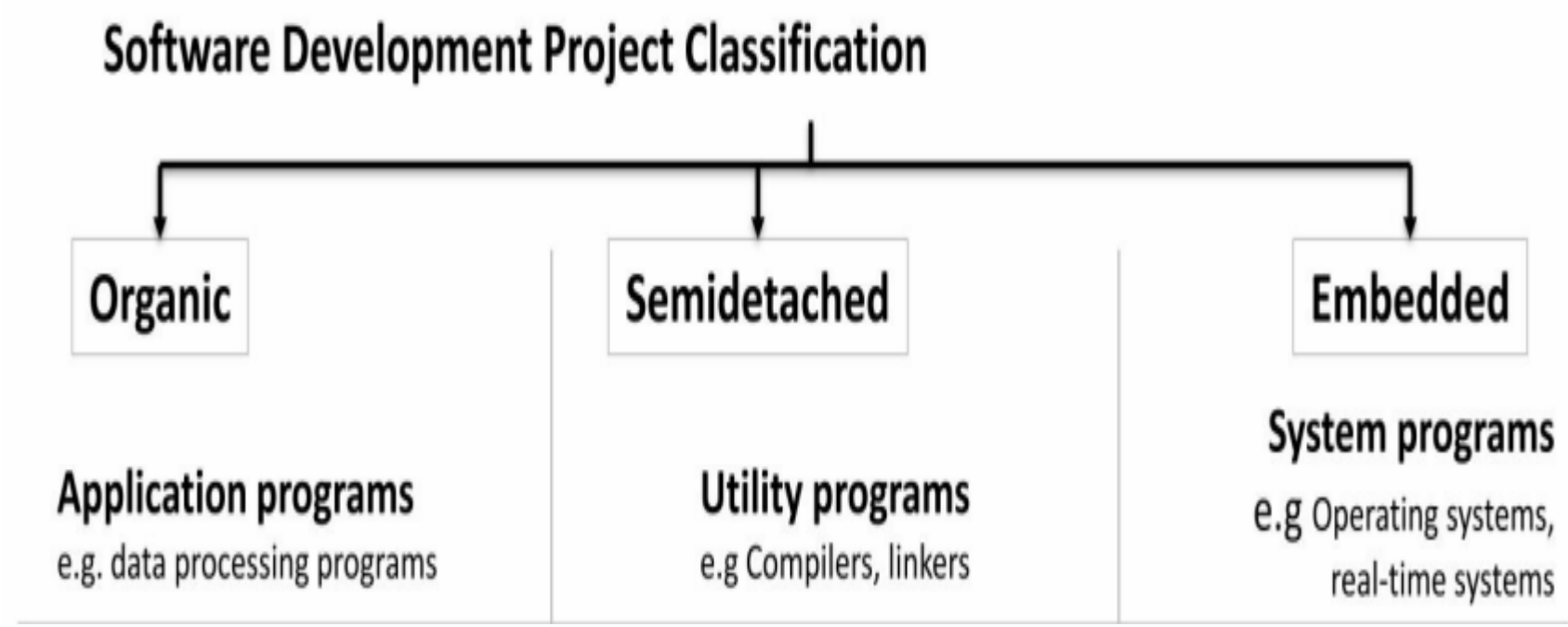
$$\text{Tdev} = b_1 \times (\text{Effort})^{b2} \text{ Months}$$

Tdev is the estimated time to develop the software, expressed in months
b1, b2 → constants based on project type

Staffing (P)

$$P = \frac{E}{D}$$

Software Development Project



Software Development Project

Model	Project Size	Nature of Project	Innovation	Dead Line	Development Environment
Organic	Typically 2-50 KLOC	Small Size Project, Experienced developers in the familiar environment, E.g. Payroll, Inventory projects etc.	Little	Not Tight	Familiar & In-house
Semi Detached	Typically 50-300 KLOC	Medium Size Project, Medium Size Team, Average Previous Experience, e.g. Utility Systems like Compilers, Database Systems, editors etc.	Medium	Medium	Medium
Embedded	Typically Over 300 KLOC	Large Project, Real Time Systems, Complex interfaces, very little previous Experience. E.g. ATMs, Air Traffic Controls	gnificant equired	Tight	Complex hardware & customer Interfaces

Basic COCOMO Model Formula

Project Types and Constants

Project	A1	A2	B1	B2
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Real-Time Example – Basic COCOMO

Project: Online Ticket Booking Website
Estimated Size: 50 KLOC
Project Type: Organic

Step 1: Calculate Effort

$$E = 2.4 \times (50)^{1.05}$$

Compute:

$$50^{1.05} \approx 60.8$$

$$E = 2.4 \times 60.8 = 146 \text{ person-months}$$

Effort = 146 Person-Months

Real-Time Example – Basic COCOMO

Step 2: Calculate Development Time

$$D = 2.5 \times (146)^{0.38}$$

Compute:

$$146^{0.38} \approx 6.6$$

$$D = 2.5 \times 6.6 = 16.5 \textit{ months}$$

Time = 16.5 months

Step 3: Calculate Staffing

$$P = \frac{145}{16.5} \approx 8.7$$

Team Required ≈ 9 Developers

Intermediate COCOMO

Formula

$$E = a1 \times (KLOC)^{a2} \times EAF$$

Where:
EAF = Effort Adjustment Factor (product of cost drivers)
Cost drivers include:
Product complexity
Required reliability
Team experience
Tools used

Detailed / Complete COCOMO Model

- The model incorporates all qualities of both Basic COCOMO and Intermediate COCOMO strategies on each software engineering process.
- The whole software is divided into different modules, and then apply COCOMO in different modules to estimate effort and then sum the effort.

Detailed / Complete COCOMO Model

The Six phases of detailed COCOMO are:

1. Planning and requirements
2. System design
3. Detailed design
4. Module code and test
5. Integration and test
6. Cost Constructive model

(Useful for large enterprise-level projects like banking systems.)

FPA – Function Point Analysis

Function Point Analysis is a **standard method to measure the functionality of software** from the user's perspective.

Unlike COCOMO, FPA does **NOT** depend on lines of code.

It measures:

- Inputs
- Outputs
- User interactions
- Files
- External interfaces

FPA – Function Point Analysis

Why Do We Need FPA?

1. To Make Estimation Technology independent
2. To Measure Software Size
3. To Improve Estimation Accuracy

4. To Support Better Project Planning
   With a clear size measure, teams can plan:
   Effort, staffing, timelines, budgets

5. To Strengthen Requirement Understanding
   - FPA forces analysts to clearly identify:
     - inputs
     - Outputs
     - data stores
     - user interactions

FPA – Function Point Analysis

Elements Counted in FPA

Element	Meaning	Example
EI – External Inputs	Data entering system	Login form, registration, order entry
EO – External Outputs	Data leaving system	Reports, invoices, emails
EQ – External Queries	Input + output	Search feature
ILF – Internal Logical Files	Internal data tables	User table, product table
EIF – External Interface Files	Data from external systems	Payment gateway data

FPA Calculation Steps

Count Unadjusted Function Points (UFP)

Weight Table

1. External Inputs (EI)

Complexity	Weight
Low	3
Medium	4
High	6

2. External Outputs (EO)

Complexity	Weight
Low	4
Medium	5
High	7

3. External Queries (EQ)

Complexity	Weight
Low	3
Medium	4
High	6

FPA Calculation Steps

Count Unadjusted Function Points (UFP)

Weight Table

4. Internal Logical Files (ILF)

Complexity	Weight
Low	7
Medium	10
High	15

5. External Interface Files (EIF)

Complexity	Weight
Low	5
Medium	7
High	10

Steps to Calculate– Function Point

Step 1: Calculate Unadjusted Function Points (UFP)

Item: Count*Complexity

$$UFP = \text{Sum of 5 items}$$

Step 2: Calculate Value Adjustment Factor (VAF)

Step 3: Calculate VAF

$$VAF = 0.65 + (0.01 \times GSC)$$

Step 4: Calculate Final Function Points (FP)

$$FP = UFP \times VAF$$

Real-Time Example – Function Point Calculation

Project: Online Shopping Website
Given Counts

Item	Count	Complexity
External Inputs (EI)	10	Medium
External Outputs (EO)	6	High
External Queries (EQ)	8	Low
ILF	4	Medium
EIF	3	Low

Real-Time Example – Function Point Calculation

Step 1: Calculate UFP

EI: 10 inputs × weight 4 = 40
EO: 6 outputs × weight 7 = 42
EQ: 8 queries × weight 3 = 24
ILF: 4 files × weight 10 = 40
EIF: 3 files × weight 5 = 15

Total UFP

$$UFP = 40 + 42 + 24 + 40 + 15 = 161$$

Real-Time Example – Function Point Calculation

Step 2: Calculate Value Adjustment Factor (VAF)
There are **14 General System Characteristics (GSCs)**:

1. Data communications
2. Performance
3. Transaction rate
4. Reusability
5. Security
6. Complex processing
7. Distributed processing
8. User efficiency
9. Backup & recovery
10. Data complexity
11. Maintainability
12. Training
13. Installation
14. On-line data entry

Each is rated **0–5**

Suppose total GSC score = 40

Real-Time Example – Function Point Calculation

Step 3: Calculate VAF

$$VAF = 0.65 + (0.01 \times GSC)$$

$$VAF = 0.65 + (0.01 \times 40) = 0.65 + 0.40 = 1.05$$

Step 4: Calculate Final Function Points (FP)

$$FP = UFP \times VAF$$

$$FP = 161 \times 1.05 = 169.05$$

Final Function Points = 169

Real-Time Example – Function Point Calculation

Function Point Calculation Example 2

Study of requirement specification for a project has produced following results

Need for **7 inputs**, **10 outputs**, **6 inquiries**, **17 files** and **4 external interfaces**

Input and **external interface function point** attributes are of **average complexity** and all **other function points** attributes are of **low complexity**

Determine **adjusted function points** assuming complexity **adjustment value is 32**.

Project Scheduling & Tracking

Scheduling Principles

- Compartmentalization
- Interdependency
- Time Allocation
- Effort Validation
- Define Responsibilities
- Define Outcomes
- Define Milestones

Project Scheduling & Tracking

Compartmentalization
Breaking a project into smaller, manageable parts.
Example: Dividing a software project into modules like login, dashboard, and reports.

Interdependency
When one task or module depends on another to function or complete.
Example: The payment module depends on the user authentication module.

Project Scheduling & Tracking

Time Allocation
Assigning a fixed amount of time to each project activity.
Example: Allocating 2 weeks for design and 3 weeks for coding.

Effort Validation
Checking whether the estimated effort matches the actual work done.
Example: Comparing planned 100 hours of coding with actual hours spent.

Define Responsibilities
Clearly assigning tasks to specific team members.
Example: One developer handles backend, another handles frontend.

Project Scheduling & Tracking

Define Outcomes
Clearly stating what result is expected from a task or phase.
Example: Outcome of testing phase is a bug-free, stable software build.

Define Milestones
Setting key checkpoints to track project progress.
Example: Completion of design document by the end of week 2.

Scheduling methods

1. **Program Evaluation and Review Technique (PERT)**
2. **Critical Path Method (CPM)**

Both **PERT** and **CPM** provide quantitative tools that allow you to:

Determine the critical path—the chain of tasks that determines the duration of the project

Establish "most likely" time estimates for individual tasks by applying statistical models

Calculate "boundary times" that define a "time window" for a particular task

Effort Distribution

General guideline: **40-20-40 rule**

40% or more of all effort allocated to **analysis and design tasks**
20% of effort allocated to **programming**
40% of effort allocated to **testing, Deployment & Maintenance**

Although most software organizations encounter the following projects types:

1. Concept Development
2. New Application Development
3. Application Enhancement
4. Application Maintenance
5. Reengineering

Risk analysis & Management

Risk Analysis and Risk Management are systematic processes used to identify, assess, and control threats that could negatively impact an organization, project, or system.

These threats may come from financial uncertainty, legal liabilities, strategic failures, natural disasters, or cybersecurity incidents.

Predictable Risk Categories

- Project risks
- Technical risks
- Business risks

Sub-categories of Business risks

- Market risk
- Strategic risk
- Sales risk
- Management risk
- Budget risk

Steps for Risk Management

1. **Identify** possible **risks** and recognize what can go wrong

2. **Analyze** each **risk** to estimate the probability that it will occur and the impact (i.e., damage) that it will do if it does occur

3. **Rank** the **risks** by probability and impact. Impact may be negligible, marginal, critical, and catastrophic.

4. **Develop** a contingency **plan** to **manage** those **risks** having high probability and high impact

RMMM

RMMM - Mitigation, Monitoring, and Management
An effective strategy for dealing with risk must consider three issues

Risk mitigation (i.e., avoidance, try to stop the risk or reduce its effect)

Risk monitoring

Risk management (Handle the risk if it actually happens)

RMMM

Risk Mitigation
Actions taken **in advance** to reduce the probability or impact of a risk.
Goal: *Prevent the risk or reduce its damage*

Risk Monitoring
Continuous tracking of identified risks to detect early warning signs.
Goal: *Know when a risk is about to happen*

Risk Management
Actions taken **when a risk occurs** (contingency plans).
Goal: *Control damage and recover quickly*

Risk Estimation

Risk estimation is a part of risk assessment that **measures the probability and impact** of identified risks to determine their severity. It provides a **numerical or qualitative value** to express how serious a risk is.

Uses descriptive scales.
Common Scale
Likelihood: 1 (Low) to 3 (High)
Impact: 1 (Low) to 3 (High)
Risk Score = Likelihood × Impact

Risk Estimation

Likelihood	Description
1	Unlikely
2	Possible
3	Likely

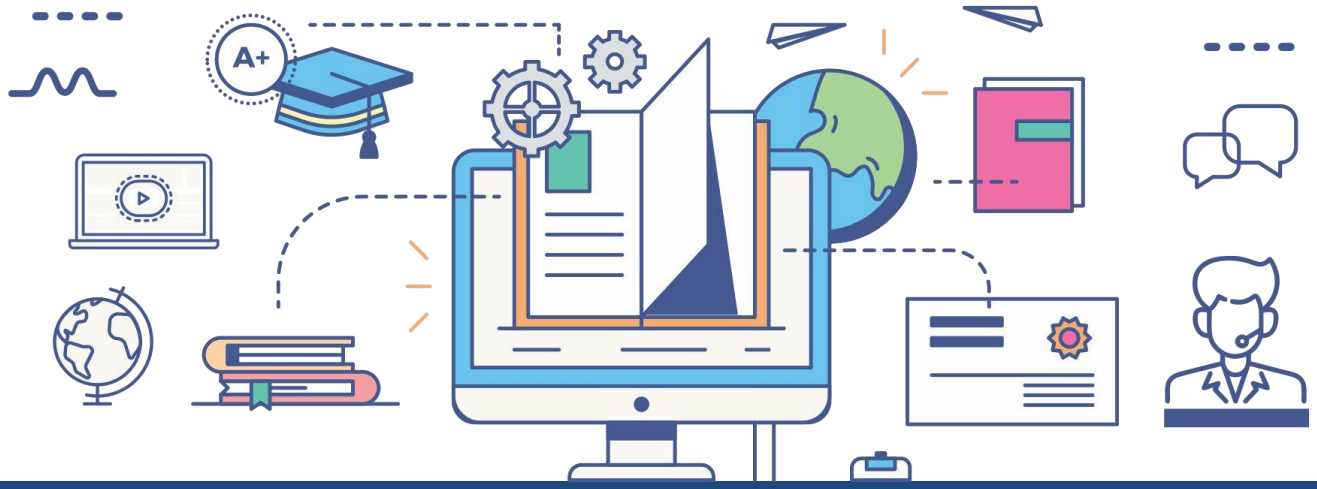
Impact	Description
1	Minor
2	Moderate
3	Major

Risk Estimation

Example (Qualitative)
A data breach risk:
•Likelihood = 3
•Impact = 3

Calculate the Risk and say the risk is High or Low

DIGITAL LEARNING CONTENT



Parul® University

www.paruluniversity.ac.in