Introduction to Software & Software Engineering

- - Lecture 1 - UG Class
- - Definition, Types, Characteristics, Attributes, Costs

What is Software?

- - Executable code + libraries + documentation
- - Difference between Program & Software Product

Types of Software

- - System Software
- - Application Software
- - Embedded Software
- - Engineering/Scientific Software
- - Web & Mobile Apps

Characteristics of Good Software

- - Operational: Functionality, Reliability, Efficiency
- - Transitional: Portability, Reusability
- - Maintainability: Modularity, Scalability

Operational Characteristics

- These relate to how well the software performs during actual use.
- **Functionality**
  - The ability of software to meet the **intended purpose** and perform the **required tasks correctly**.
  - Example: A billing software should accurately calculate totals and generate invoices.
- **Reliability**
  - The software should work **consistently without failure** under expected conditions.
  - Example: An ATM software must reliably process transactions without errors or crashes.
- **Efficiency**
  - The software should use **minimum resources** (CPU, memory, network) and respond **quickly**.
  - Example: A mobile app should load fast and not drain too much battery or data.

Transitional Characteristics

These relate to how easily the software can be **moved or adapted** to different environments.

- **Portability**
  - The ability to **run on different hardware or operating systems** with minimal changes.
  - Example: A web application that works smoothly on Windows, Mac, and Linux.
- **Reusability**
  - Parts of the software (like modules or libraries) should be **usable in other projects**.
  - Example: A login authentication module being reused in multiple web apps.

Maintainability Characteristics

These relate to how easily the software can be **modified or extended** after deployment.

- **Modularity**
  - The software should be divided into **independent modules** so that changes in one don’t affect others.
  - Example: In an e-commerce app, product listing and payment gateway can be separate modules.
- **Scalability**
  - The software should handle **increasing workloads** (more users, more data) without performance degradation.
  - Example: A social media platform should scale easily from thousands to millions of users.

Software Attributes - The 'ilities'

- - Functional correctness
- - Usability & Maintainability
- - Efficiency & Performance
- - Reliability & Security
- - Scalability & Adaptability

What is Software Engineering?

- - Rajib Mall: Engineering discipline for reliable software
- - IEEE: Systematic, disciplined approach
- - Layers: Process -> Methods -> Tools

Why Software Engineering?

- - Manage complexity
- - Improve quality
- - Control costs & schedules
- - Handle changing requirements

Software Engineering Costs

- - Software cost > Hardware cost
- - Drivers: Complexity, maintenance, changes
- - Models like COCOMO help estimate costs

Software cost > Hardware cost

- **Hardware** is a **one-time physical investment** (e.g., buying a server). Once purchased, its cost is mostly fixed.
- **Software**, on the other hand, involves:
  - **Development cost** → designing, coding, testing, documentation
  - **Maintenance cost** → fixing bugs, updating features, adapting to new environments
  - **Evolution cost** → meeting new business needs, scaling for more users
- ☞ **Over time, maintenance & upgrades cost much more than the initial hardware.**

Key **Cost Drivers** in Software Engineering

- **Complexity**
  - More complex systems require **more effort** to design, test, and maintain.
  - Example: A simple calculator app costs less than a banking system.
- **Maintenance**
  - Software **changes frequently** (bug fixes, security updates, new features).
  - Studies show **70–80% of total cost is spent after deployment**.
- **Changing Requirements**
  - Business needs evolve → requires **rework** and **adaptation**, increasing cost.

How to **Estimate Software Costs?**

Models like **COCOMO (Constructive Cost Model)** are used.

- They consider:
  - **Size of software** (measured in KLOC = thousand lines of code)
  - **Complexity level** (simple, intermediate, embedded)
  - **Team experience & tools**
  - Output → **Effort (person-months)** and **Time (schedule)** required → helps calculate cost.

Managing Costs

- - Use modular design
- - Promote reuse
- - Employ proper process & CASE tools

Summary & Takeaways

- - Software definition & types
- - Key attributes & 'ilities'
- - Software engineering definition & cost factors

Q&A / Discussion

- - Which attribute is hardest to ensure?
- - Can software ever be 'complete'?