SAD(MID-Note)

CHAPTER - 01

Definition of System?

“A system is a group of interrelated components that work together in an organized way to achieve a specific goal or desired result.”

Types of Information Systems?

1. **Transaction Processing System (TPS)**
    - Handles day-to-day routine transactions.
    - Example: ATM transactions, online ticket booking, payroll system.
2. **Office Automation System (OAS)**
    - Helps in communication, data storage, and document management.
    - Example: MS Office, email systems, Google Workspace.
3. **Knowledge Work System (KWS)**
    - Supports knowledge creation and integration.
    - Example: CAD (Computer-Aided Design), research databases.
4. **Management Information System (MIS)**
    - Provides summarized reports for managers to make decisions.
    - Example: Sales reports, inventory control systems.
5. **Decision Support System (DSS)**
    - Assists in decision-making using data models and analysis tools.

- Example: Forecasting tools, financial planning systems.

6. **Expert System (ES)**
    - Mimics human experts by using AI rules and reasoning.
    - Example: Medical diagnosis systems (MYCIN), troubleshooting systems.
7. **Group Decision Support System (GDSS)**
    - Helps groups/teams in collaborative decision-making.
    - Example: Video conferencing with decision software, brainstorming tools.
8. **Executive Support System (ESS)**
    - Provides top executives with easy access to critical information for strategic decisions.
    - Example: Dashboards with KPIs, corporate performance monitoring systems.

Need for Systems Analysis and Design

1. **Cost of Errors**
    - Fixing faults **after software delivery** is up to *100x more expensive* than fixing them in the **analysis or design phase**.
    - Early analysis prevents costly mistakes later.
2. **Proper Planning**
    - Installing a system without proper planning often leads to:
        - User dissatisfaction
        - Frequent disuse or abandonment of the system
3. **User Involvement**
    - Users must be involved throughout the process to ensure the system meets their needs.
    - SAD ensures **requirements are gathered correctly** and users are satisfied.

4. **Efficient Resource Utilization**
    - Helps avoid **wastage of time, money, and effort** by defining scope and objectives early.
5. **Improved Quality of Systems**
    - Produces **reliable, flexible, and scalable** systems.
    - Reduces chances of failure after deployment.
6. **Adapting to New Technologies**
    - With rapidly changing tech, SAD helps organizations **upgrade and integrate new technologies** effectively.

Roles of the Systems Analyst

A **systems analyst** is the key person who studies an organization's current system, identifies problems, and designs solutions. To do this, the analyst must be able to work with **people of all backgrounds** as well as be experienced in working with computers.

The **three primary roles** of a systems analyst are:

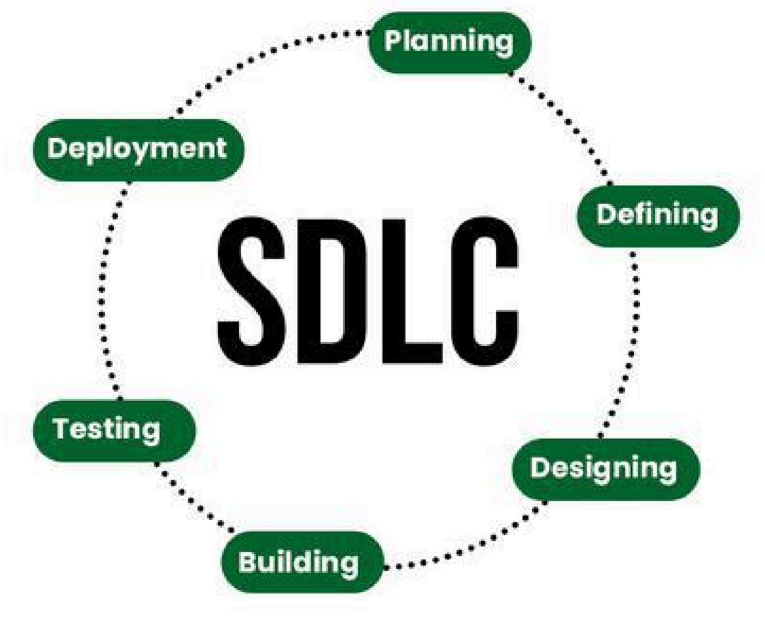
1. **Consultant**
    - Acts as an advisor to management or users.
    - Provides expertise to solve problems or improve processes.
    - Example: Suggesting the best technology for automating payroll.
2. **Supporting Expert**
    - Provides technical assistance and specialized knowledge.
    - Helps users understand what technology can (and cannot) do.
    - Example: Guiding the team on database design or network setup.
3. **Agent of Change**

- Introduces and manages changes in the organization through new systems.
- Ensures smooth transition from the old system to the new one.
- Example: Training staff and helping them adapt to a new ERP system.

Software Development Life Cycle (SDLC)

Definition:

SDLC is a structured process used to design, develop, test, and deploy high-quality software that meets user requirements.



Stages of SDLC

1. **Planning & Requirement Analysis**
    - Collect inputs from customers, sales, and market surveys.

- Assess feasibility (technical & economic).
- Forms the foundation for project planning.

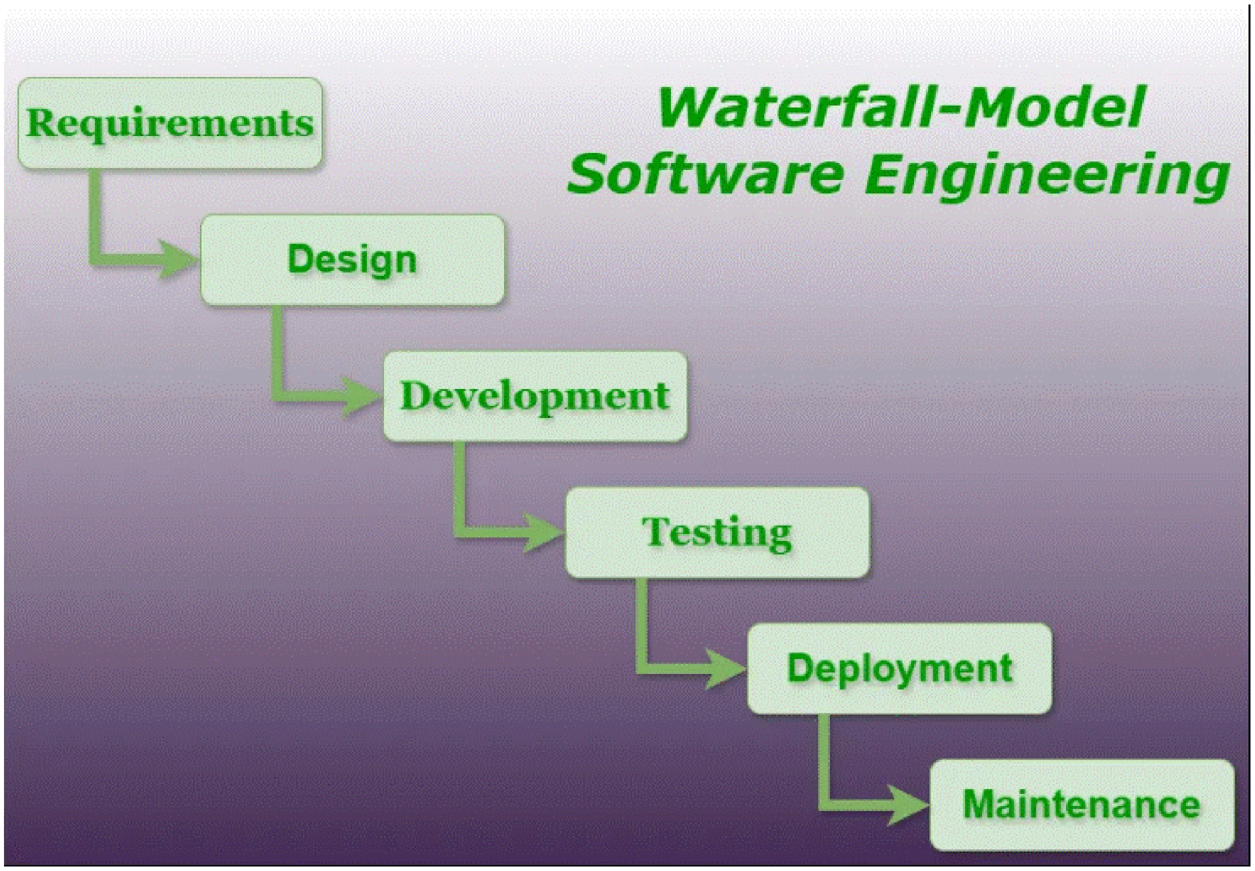
2. **Defining Requirements**
   - Document all software requirements in SRS (Software Requirement Specification).
   - Get approval from customers and stakeholders.
3. **Designing Architecture**
   - Create system architecture and detailed designs (HLD & LLD).
   - Select the best design in the Design Document Specification (DDS).
4. **Development**
   - Convert design into actual code using programming languages (C/C++, Java, Python, etc.).
   - Perform unit testing for each module.
5. **Testing & Integration**
   - Test individual modules (unit testing) and combined modules (integration testing).
   - Perform system testing to verify against requirements.
   - Prepare documentation and provide training to users.
6. **Deployment & Maintenance**
   - Deploy software to the live environment in phases.
   - Collect feedback, fix bugs, and enhance the system (corrective, perfective, adaptive maintenance).

Exam Tip:

- If short answer: just write **Planning → Requirements → Design → Development → Testing → Deployment & Maintenance** with 1–2 lines each.
- If long answer: use the above bullet points + small intro and conclusion.

What is the SDLC Waterfall Model?

The **Waterfall Model** is a traditional software development model where each phase is completed fully before moving to the next. It follows a **linear and sequential approach**, making it simple but less flexible compared to Agile.



Waterfall Model-Software Engineering

Phases of Waterfall Model

1. **Requirements Analysis and Specification**
    - Collect and document customer requirements in detail (SRS – Software Requirement Specification).
    - Ensures clarity on what the software must do.
2. **Design**

- Convert requirements into a blueprint for coding.
- **High-Level Design (HLD):** Defines system architecture and major components.
- **Low-Level Design (LLD):** Gives details of each module for coding.

3. **Development (Implementation)**
   - Actual coding of modules based on design.
   - Unit testing is done to ensure each module works correctly.
4. **Testing and Deployment**
   - **Integration Testing:** Combine modules and test them together.
   - **System Testing:** Verify the complete system against requirements.
   - **Deployment:** Release the software to users and ensure smooth installation/training.
5. **Maintenance**
   - Longest and most costly phase (around 60% effort).
   - **Corrective:** Fix bugs missed earlier.
   - **Perfective:** Add new features or improve performance.
   - **Adaptive:** Modify system for new platforms or environments.

Conclusion

The Waterfall Model is easy to understand and manage, making it suitable for small projects with clear, fixed requirements. However, it is **less flexible** since once a phase is completed, going back is very costly and difficult.

Advantages:

1. Simple and easy to understand.
2. Clear structure: each phase has defined start and end.
3. Easy to manage due to documentation and milestones.

4. Works well for small projects with fixed requirements.

Disadvantages:

1. Inflexible — difficult to go back to a previous phase.
2. Not suitable for projects with changing requirements.
3. Testing happens late, so errors may be costly.
4. User feedback is delayed until the end.

What is Agile Model?

The **Agile Model** is an iterative software development approach where the project is divided into small cycles (called iterations). Each iteration produces a working product that can be tested and improved based on user feedback. Agile focuses on flexibility, continuous delivery, and customer involvement.

Steps in the Agile Model

The Agile Model is a combination of iterative and incremental process models. The phases involve in **Agile (SDLC) Model** are:

1. **Requirement Gathering**
2. **Design the Requirements**
3. **Construction / Iteration**
4. **Testing / Quality Assurance**
5. **Deployment**
6. **Feedback**

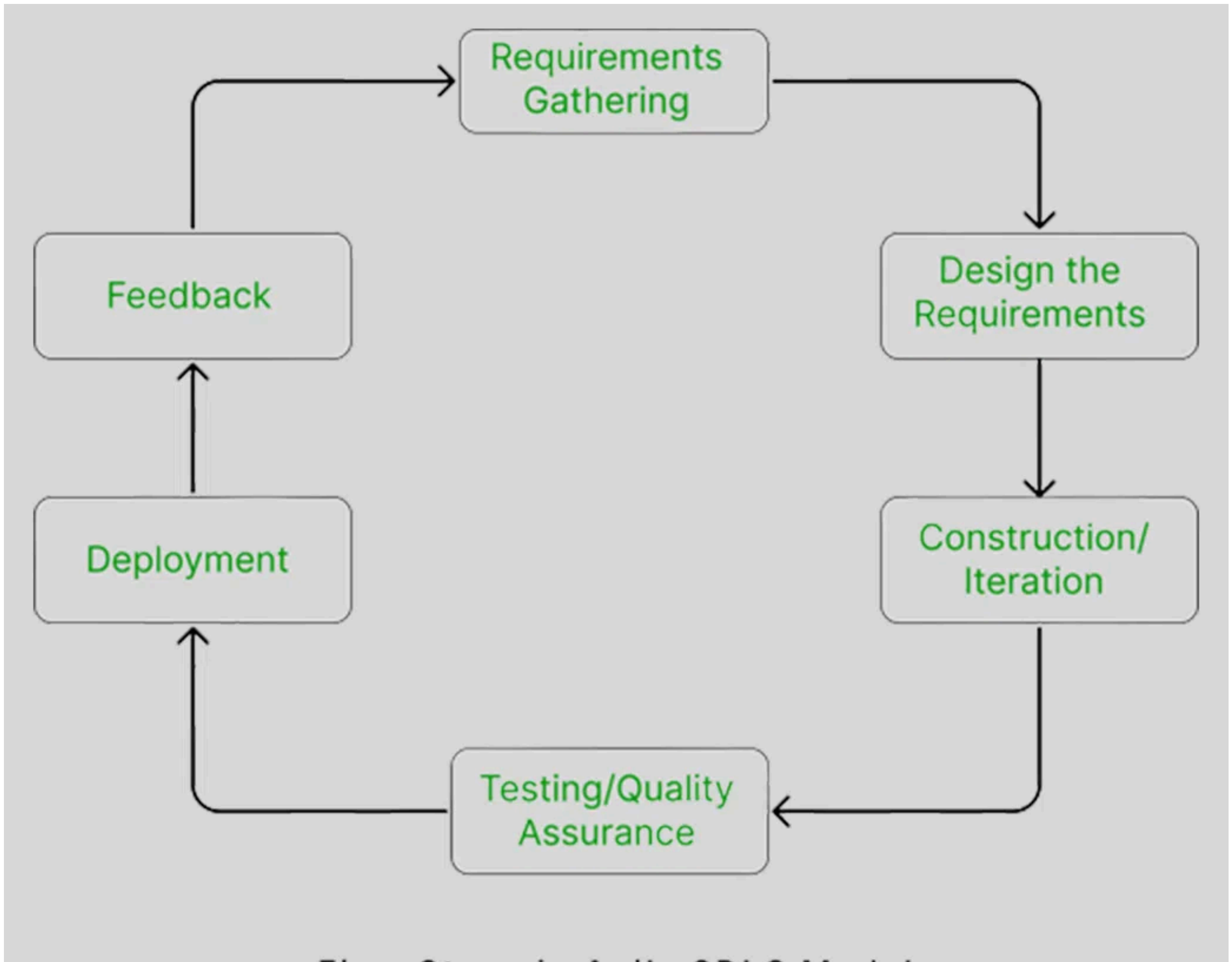


Fig:- Steps in Agile SDLC Model

Agile Model Steps

Phases of Agile Model

1. **Requirement Gathering**
    - Meet customers, identify needs and goals.
    - Estimate time and cost.
    - Check technical and economic feasibility.
2. **Design the Requirements**
    - Create overall system design and UML diagrams.
    - Prepare wireframes (UI blueprints).

- Build simple prototypes for feedback.

3. **Construction / Iteration**
    - Develop features in short cycles (1–4 weeks).
    - Perform coding and integration.
    - Deliver a working version of the software after each cycle.
4. **Testing / Quality Assurance**
    - **Unit Testing** → small parts of code.
    - **Integration Testing** → combined modules.
    - **System Testing** → full system against requirements.
5. **Deployment**
    - Release the working product to end users.
    - Provide updates and continuous improvements.
6. **Feedback**
    - Collect feedback from users and stakeholders.
    - Fix bugs and improve features.

Advantages:

1. Flexible — adapts to changing requirements.
2. Continuous delivery of working software.
3. High customer involvement and satisfaction.
4. Early detection and fixing of errors.
5. Encourages teamwork and collaboration.

Disadvantages:

1. Less predictable due to changing scope.
2. Requires experienced and committed team.
3. Documentation may be minimal.

4. Harder to manage large projects without proper planning.

Object-Oriented Design (OOD) – Exam Summary

Definition:

- OOD transforms the **analysis model (from OOA) into a detailed design model** ready for implementation.
- Specifies **how objects are structured, interact, and perform responsibilities**.

Key Aspects of OOD

1. **Data Organization of Attributes**
    - Specifies **types and structure of data** in each object.
2. **Procedural Description of Operations**
    - Details **steps/processes for each method or operation** of an object.

Object-Oriented Design Pyramid (4 Layers)

1. **Subsystem Layer** – Represents subsystems that meet **user requirements** and technical needs.
2. **Class & Object Layer** – Shows **class hierarchies**, generalization/specialization, and individual objects.
3. **Message Layer** – Defines **object interactions**, method calls, and flow of control.
4. **Responsibilities Layer** – Specifies **behavior of objects** and their **responses to messages**.

When to Choose Each Methodology

Methodology	When to Choose
SDLC	• Systems already developed/documented in SDLC • Need to document each step • Management prefers structured approach • Adequate resources & time available • Communication of system is important
Agile	• Project champion for agile exists • Need quick development in dynamic environment • Rescue situations (failed systems) • Customer accepts incremental improvements • Executives & analysts support agile
Object-Oriented (OOSAD)	• Problem fits into classes/objects • Organization supports UML • System can be built gradually • Reuse of code possible • Can tackle difficult problems first
Waterfall	• Requirements are clear & fixed • Project is small and short-term • Technology is well-known & stable • Quality is top priority (time can be longer) • Low risk of scope changes

CHAPTER 2

Levels of Management in System Analysis & Design

1. **Top-Level Management (Strategic Level)**

- **Role**: Long-term planning, policy-making, overall direction of the organization.
- **Focus**: External environment, future growth, investments, competition.
- **Systems Used**: **Executive Support Systems (ESS)**, Decision Support Systems (DSS).
- **Example**: Deciding whether to expand into a new market or launch a new product line.

---

2. **Middle-Level Management (Tactical Level)**

- **Role**: Converts top management's strategies into plans & monitors departments.

- **Focus**: Coordination, resource allocation, project planning, performance evaluation.
- **Systems Used**: **Management Information Systems (MIS)**, DSS.
- **Example**: A sales manager preparing monthly sales reports and planning targets.

---

3. **Lower-Level Management (Operational Level)**

- **Role**: Day-to-day operations, supervising employees, ensuring tasks are done.
- **Focus**: Routine activities, efficiency, accuracy, short-term goals.
- **Systems Used**: **Transaction Processing Systems (TPS)**.
- **Example**: Recording customer orders, payroll processing, tracking daily production.

✅ Summary Table for Quick Exam Memory:

Level of Management	Focus	Example	Information Systems Used
Top (Strategic)	Long-term planning	Market expansion decision	ESS, DSS
Middle (Tactical)	Department coordination	Monthly performance report	MIS, DSS
Lower (Operational)	Daily operations	Payroll, billing, inventory	TPS

CHAPTER 3

Selection of Projects in System Analysis and Design

Definition:

Project selection is the process of evaluating and choosing the most suitable system development project that aligns with organizational goals, is feasible, and provides maximum benefits compared to alternative uses of resources.

Project Selection

- **Management Support** – Strong backing from top management ensures approval, resources, and successful implementation.
- **Timing of Project Commitment** – Projects should start at the right time for maximum effectiveness.
- **Contribution to Organizational Goals** – Should help achieve strategic or operational objectives.
- **Resource Practicality** – Feasible in terms of time, money, and manpower for the organization and system analyst.
- **Worthwhile Investment** – Benefits must outweigh costs and be better than alternative uses of resources

Feasibility Study

Definition:

- A feasibility study is done to determine whether a proposed system is **practical, cost-effective, and useful** for the organization before starting development.
- It helps **avoid wasting time, money, and resources** on projects that may fail.

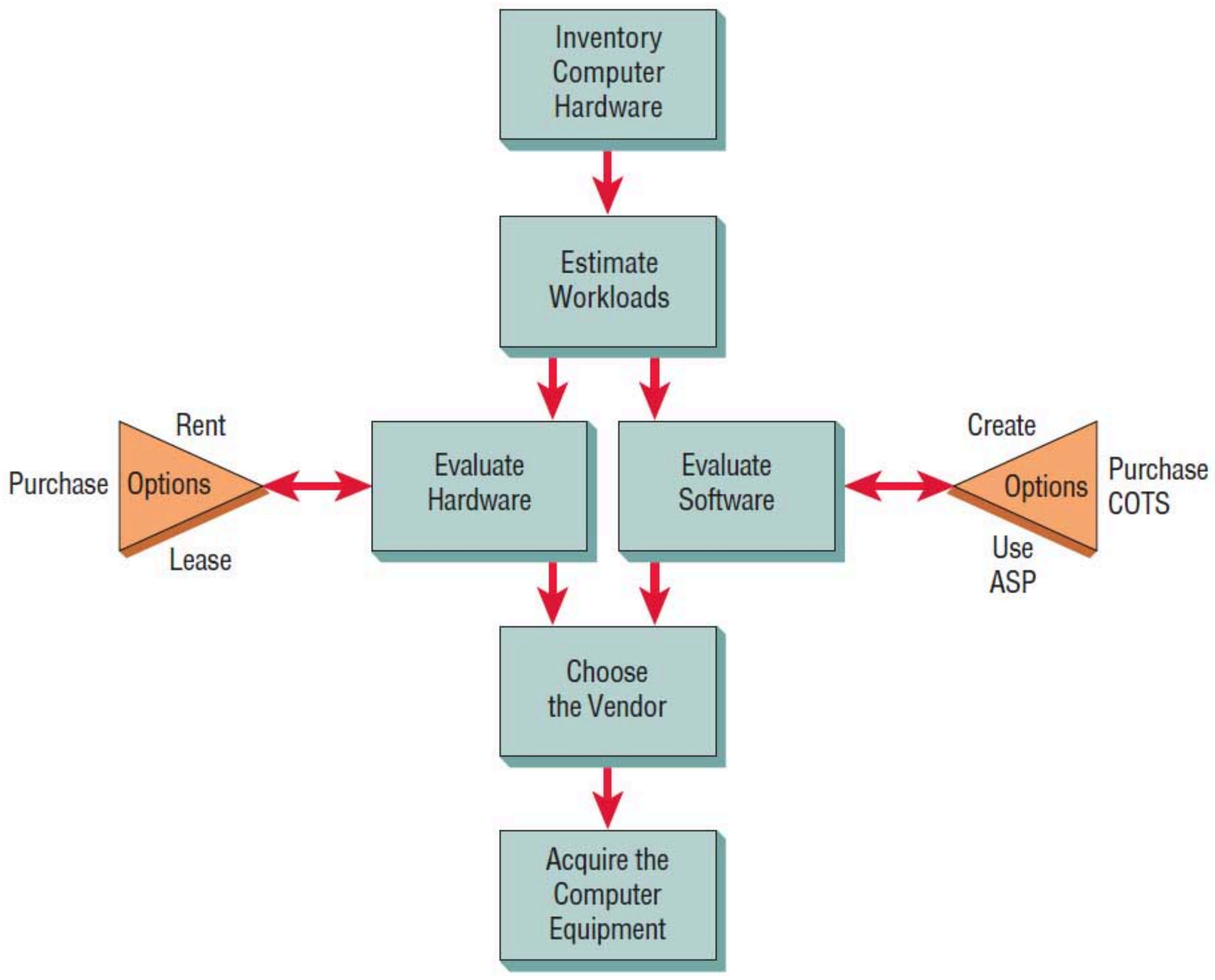
Types of Feasibility (Exam Version)

1. **Technical Feasibility**

- Checks if the organization has the **hardware, software, and technical expertise** to build and run the system.
- Ensures the technology exists and the team can implement it.
- *Example:* Can the current IT infrastructure support a new hospital management system?

2. **Economic Feasibility**
    - Evaluates if the **cost of the system** is justified by the **expected benefits**.
    - Considers development cost, hardware/software, training, and maintenance.
    - *Example:* Will automating payroll save enough money to cover software and setup costs?
3. **Operational Feasibility**
    - Checks whether the system will **function properly in the organization** and be accepted by employees.
    - Considers how easily users can adapt and whether it improves operations.
    - *Example:* Will library staff be able to use the new automated book issuing system effectively?

Steps in Choosing Hardware and Software



Software Alternatives

Definition:

- Software alternatives are **different options available to meet the software requirements** of an organization.
- Selecting the right software depends on **cost, functionality, compatibility, and user needs**.

Types of Software Alternatives

1. **Custom-Built Software (In-House Development)**
    - Developed **specifically for the organization**.
    - Fully **tailored to meet business needs**.

- *Pros:* Exact fit, flexible, scalable.
- *Cons:* Expensive, time-consuming.

2. **Off-the-Shelf Software (Commercial Software)**
    - Pre-built software available for **general use**.
    - *Pros:* Ready to use, cheaper, supported by vendor.
    - *Cons:* May not fit exact needs, less flexible.
3. **Hybrid Approach**
    - Combines **off-the-shelf software with custom development**.
    - *Pros:* Balance of cost and customization.
    - *Cons:* Integration can be complex.

Guidelines for Evaluating Software

Definition:

- Evaluating software is the process of **assessing different software options** to determine which one best meets the organization's requirements.

Key Guidelines

1. **Functionality**
    - Check if the software **meets all business requirements**.
    - Ensure it has the **necessary features** for current and future needs.
2. **Usability**
    - Evaluate how **easy it is for users to learn and operate** the software.
    - Consider **user interface, help features, and documentation**.
3. **Compatibility**
    - Ensure software works with **existing hardware, operating systems, and other applications**.

4. **Reliability and Performance**
    - Assess software **stability, speed, and error handling**.
    - Check for **uptime, crash frequency, and recovery options**.
5. **Cost**
    - Consider **purchase, licensing, implementation, training, and maintenance costs**.
    - Evaluate **total cost of ownership (TCO)**.
6. **Vendor Support and Maintenance**
    - Check if the vendor provides **technical support, updates, and patches**.
    - Consider **long-term availability** of support.
7. **Security**
    - Assess **data protection, user access control, and compliance** with regulations.

Identifying Benefits and Costs

Definition:

- Before implementing a system, organizations need to **identify the benefits they will gain** and **the costs they will incur**.
- Helps in **deciding whether the system is worthwhile**.

Identifying Costs

1. **Development Costs**
    - Hardware, software, programming, and testing.
2. **Operational Costs**
    - Maintenance, training, electricity, and support staff.
3. **Indirect Costs**

    - Productivity loss during system transition, downtime, or errors.

Tangible and Intangible Benefits and Costs

1. Tangible Benefits

- **Definition:** Benefits that can be **measured in monetary terms**.
- **Examples:**
    - Increased revenue or sales
    - Reduced labor costs
    - Faster processing or production
    - Reduced material wastage

---

2. Intangible Benefits

- **Definition:** Benefits that are **not easily measured in money** but improve the organization.
- **Examples:**
    - Improved customer satisfaction
    - Better employee morale
    - Enhanced brand image or reputation
    - Better decision-making

---

3. Tangible Costs

- **Definition:** Costs that can be **quantified in money**.
- **Examples:**
    - Hardware and software purchase
    - Training costs
    - Maintenance and support fees

- Salaries of additional staff

---

4. Intangible Costs

- **Definition:** Costs that **cannot be directly measured in money** but affect the organization.
- **Examples:**
  - Employee frustration or dissatisfaction
  - Downtime during system implementation
  - Loss of customer goodwill due to errors
  - Resistance to change

Work Breakdown Structure (WBS)

Definition:

- WBS is a **hierarchical decomposition of a project into smaller, manageable components or tasks**.
- It helps **organize work, assign responsibilities, and plan project schedules**.

Key Points

1. **Hierarchical Structure**
   - Project is broken down from **major deliverables → sub-deliverables → tasks → subtasks**.
2. **Purpose**
   - Simplifies **project planning, monitoring, and control**.
   - Helps in **resource allocation and cost estimation**.
3. **Benefits**
   - Makes complex projects **manageable**.
   - Ensures **nothing is overlooked**.

- Helps **track progress** easily.

BYOD (Bring Your Own Device)

Definition:

- BYOD is a policy where employees are **allowed to use their personal devices** (laptops, smartphones, tablets) for work purposes instead of only company-provided devices.

Benefits of BYOD

1. **Cost Savings**
    - Reduces company expenses on purchasing and maintaining devices.
2. **Increased Productivity**
    - Employees work on devices they are familiar with, making them faster and more efficient.
3. **Flexibility & Mobility**
    - Employees can work from anywhere, supporting remote work or flexible schedules.
4. **Employee Satisfaction**
    - Using personal devices makes employees more comfortable and satisfied.
5. **Latest Technology**
    - Employees often use modern devices, so the organization benefits from up-to-date tech without extra cost.

Cost-Benefit Analysis (CBA)

Definition:

- Cost-Benefit Analysis is the process of **comparing the total expected costs of a project or system with its expected benefits** to determine whether it is financially worthwhile.
- It helps organizations make informed decisions before investing in a system.

Steps in Cost-Benefit Analysis

1. **Identify Costs**
    - Development costs: hardware, software, programming, testing.
    - Operating costs: maintenance, training, electricity, support staff.
2. **Identify Benefits**
    - Tangible benefits: increased revenue, reduced labor costs, faster processing.
    - Intangible benefits: improved customer satisfaction, better decision-making, higher employee morale.
3. **Compare Costs and Benefits**
    - Determine whether the **benefits outweigh the costs**.
    - Calculate ROI (Return on Investment) if needed.

Purpose / Importance

- Helps decide if a project is **worth implementing**.
- Ensures **efficient use of resources**.
- Reduces risk of **financial loss**.

Break-Even Analysis (BEA)

Definition:

- Break-Even Analysis determines the **point at which total costs equal total revenue**, meaning the project neither makes a profit nor incurs a loss.
- It helps in **financial planning** and deciding if a project is viable.

Key Points / Steps

1. **Identify Costs**
   - Fixed costs: do not change with production (e.g., hardware, software, salaries).
   - Variable costs: change with usage/production (e.g., consumables, electricity).
2. **Calculate Break-Even Point (BEP)**
   - **BEP = Fixed Costs ÷ (Selling Price per Unit – Variable Cost per Unit)**
   - Represents the number of units or revenue needed to cover all costs.
3. **Analyze and Make Decisions**
   - If projected revenue is **above BEP**, project is profitable.
   - If below BEP, reconsider project feasibility or cost structure.

Earned Value Management (EVM)

Definition:

- EVM is a **project management technique** used to measure project performance and progress in terms of **scope, time, and cost**.
- It helps managers see if a project is **on track, ahead, or behind schedule and budget**.

Key Components

1. **Planned Value (PV)**
   - The budgeted cost for work **scheduled to be completed** by a certain date.

2. **Earned Value (EV)**
    - The **budgeted cost of actual work completed** by a certain date.
3. **Actual Cost (AC)**
    - The **actual cost incurred** for the work done by a certain date.

Key Formulas

- **Cost Variance (CV) = EV – AC** → Positive = under budget, Negative = over budget
- **Schedule Variance (SV) = EV – PV** → Positive = ahead of schedule, Negative = behind schedule

---

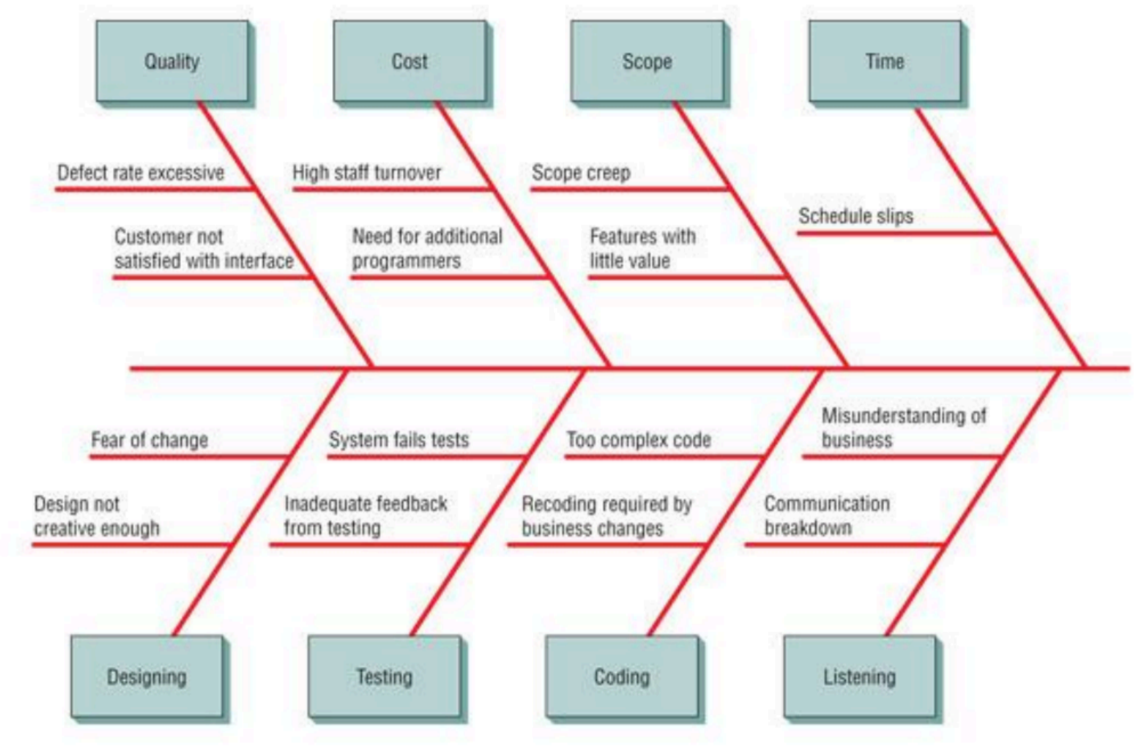
Benefits

- Provides **early warning** of schedule or budget problems.
- Helps in **controlling project costs and timelines**.
- Enables **objective performance measurement**.

✅ **Exam Tip:**

- Write **definition + 3 components + 2 formulas + 1-line benefit** — enough for 4–5 marks.
- Optional example: “If EV = $50,000, AC = $45,000 → CV = $5,000 (under budget).”

Project Risk:


Quality
Cost
Scope
Time
Defect rate excessive
High staff turnover
Scope creep
Customer not satisfied with interface
Need for additional programmers
Features with little value
Schedule slips
Fear of change
System fails tests
Too complex code
Misunderstanding of business
Design not creative enough
Inadequate feedback from testing
Recoding required by business changes
Communication breakdown
Designing
Testing
Coding
Listening

Team Management

Definition:

Team management is the process of organizing, coordinating, and leading a group of people to achieve project or organizational goals efficiently. It ensures collaboration, productivity, and motivation among team members.

Four Steps of Team Management

1. **Assembling a Team**
    - Selecting the right people with the necessary skills and expertise.
    - Ensures balance of technical and interpersonal abilities.
2. **Team Communication Strategies**

- Establishing clear methods and tools for sharing information.
- Promotes feedback, transparency, and reduces misunderstandings.

3. **Project Productivity Goals**
    - Setting measurable and realistic targets for the team.
    - Ensures timely completion and project success.
4. **Team Member Motivation**
    - Inspiring and encouraging individuals to perform their best.
    - Builds engagement, morale, and long-term commitment.