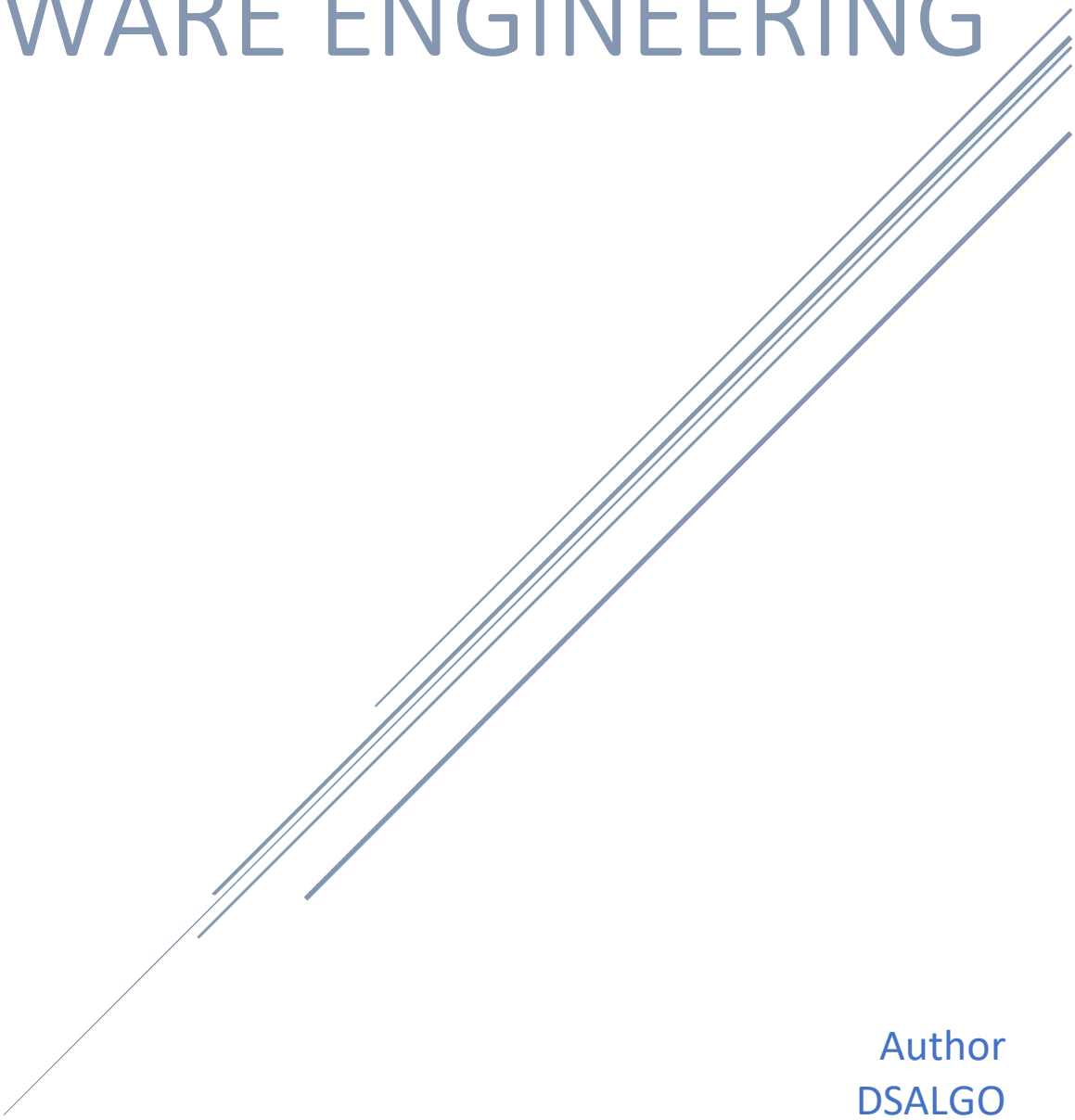
# SOFTWARE ENGINEERING

## UNIT --2



Author
DSALGO

## Requirement engineering

- Requirement engineering in software engineering is the process of eliciting, analyzing, documenting, and managing requirements for a software system.
- Here's an overview of the key aspects:
  - **Elicitation**: Gathering requirements from stakeholders.
  - **Analysis**: Refining requirements for clarity and feasibility.
  - **Documentation**: Structured documentation of requirements.
  - **Validation**: Ensuring requirements meet stakeholder needs.
  - **Management**: Managing requirements throughout the project.
  - **Communication**: Facilitating stakeholder collaboration.

## Use of requirement engineering

1. Understanding user needs.
2. Risk management.
3. Quality assurance.
4. Customer satisfaction.
5. Scope definition.
6. Cost estimation.
7. Communication and collaboration.

## requirement elicitation

- requirement elicitation is the crucial initial phase in the software development lifecycle
- encompassing the systematic gathering and discovery of requirements from stakeholders, including end-users, customers, subject matter experts, and other relevant parties.
- It involves a series of activities aimed at identifying, understanding, and documenting the needs, preferences, constraints, and expectations associated with the software to be developed.

**Methods:**

1. Interviews
2. Workshops
3. Prototyping
4. Document Analysis
5. Use Cases
6. Surveys
7. Observation
8. Focus Groups
9. Questionnaires
10. Storytelling

**Procedure:**

- **Identify Stakeholders**: Determine relevant stakeholders, including end-users, experts, etc.
- **Select Techniques**: Choose appropriate methods like interviews, surveys, workshops.
- **Conduct Activities**: Engage stakeholders to gather needs and preferences.
- **Document Requirements**: Record in structured format (docs, stories, use cases).
- **Validate Requirements**: Confirm accuracy and clarity with stakeholders.

**Problems:**

- **Incomplete Requirements**: Missing or unclear specifications.
- **Miscommunication**: Poor understanding due to communication issues.
- **Changing Requirements**: Evolving needs causing challenges in management.
- **Resistance to Change:** Reluctance to adapt to new requirements.
- **Limited Involvement:** Not enough engagement from key stakeholders.

## System documentation

- System documentation comprises a set of documents detailing the architecture, design, functionality, and usage of a software system.
- It includes documents such as system overview, architecture, design, functionality, installation and configuration guides, user manuals, API documentation, maintenance and support procedures, testing documentation, and security measures.
- It facilitates understanding, communication, collaboration, consistency, accuracy, onboarding of new team members, and future enhancements and updates.

## types of documentation

- **Requirements Documentation**: Outlines what the software needs to do, including specifications and user stories.
- **Design Documentation**: Describes how the software is structured and how its components interact, using diagrams and patterns.
- **User Documentation**: Provides instructions for users on how to use the software, including manuals and guides.
- **Technical Documentation**: Details the technical aspects of the software for developers and testers, such as APIs and code documentation.
- **Project Documentation**: Tracks the management and planning of the software project, including plans, schedules, and meeting minutes.

## Reviewing User Requirements:

- **Evaluation**: Assess completeness, clarity, consistency, and feasibility.
- **Alignment**: Ensure requirements match project objectives and stakeholders' needs.
- **Identification**: Detect gaps, ambiguities, or conflicts in documentation.

## Managing User Requirements:

- **Process Establishment**: Set up a structured management process.
- **Documentation and Tracking**: Record and monitor changes to requirements.
- **Communication**: Share changes with stakeholders and gather feedback.
- **Storage and Accessibility**: Ensure proper documentation storage and accessibility.
- **Regular Review and Update**: Periodically revisit and adjust requirements.
- **Validation**: Verify requirements' validity, relevance, and alignment with goals.

*"Effective review and management ensure the software meets stakeholders' needs, leading to project success."*

## Feasibility study

- Feasibility study is a preliminary analysis conducted to assess the practicality and viability of a proposed project or solution.
- It aims to determine whether the project is technically, economically, and operationally feasible before committing resources to its development.

## Type of feasibility study

- Technical Feasibility Study:
  - Check availability of technology and resources.
  - Assess compatibility with existing systems.
- Economic Feasibility Study:
  - Estimate costs for project implementation.
  - Evaluate potential returns on investment.
- Operational Feasibility Study:
  - Examine project practicality from operational standpoint.
  - Analyze alignment with existing business processes.

## Important 8 points for feasibility

- **Define Project Scope and Objectives:** Clearly define the purpose,
- goals, and scope of the proposed project.
- **Gather Information**: Collect relevant data and information related to the project, including technical requirements and financial data.
- **Identify Alternatives**: Explore various solutions or approaches to address the project objectives.
- **Evaluate Alternatives:** Assess the feasibility of each alternative based on technical, economic, and operational criteria.
- **Conduct Cost-Benefit Analysis:** Analyze the costs associated with each alternative and compare them to the expected benefits.
- **Make Recommendations**: Recommend the most feasible alternative based on the analysis and evaluation.
- **Prepare Feasibility Report:** Document the findings of the feasibility study in a comprehensive report.
- **Present Findings to Stakeholders**: communicate the results and recommendations of the feasibility study to stakeholders

## Different approaches for information modeling

**1.Formal Methods:**

- **Entity-Relationship Modeling (ER):** Represents data in terms of entities and relationships between them.
- **Object-Oriented Modeling (OOM):** Models data as a collection of objects, each encapsulating data and behavior.
- **Data Flow Modeling (DFM):** Focuses on representing the flow of data within a system or process.
- **Structured Analysis and Design Technique (SADT):** Uses graphical representations to model system processes, data flows, and data stores.
- **Unified Modeling Language (UML):** Provides a standardized notation for modeling software systems.
- **Formal Concept Analysis (FCA):** Represents data as a lattice structure of formal concepts derived from a set of objects and attributes.

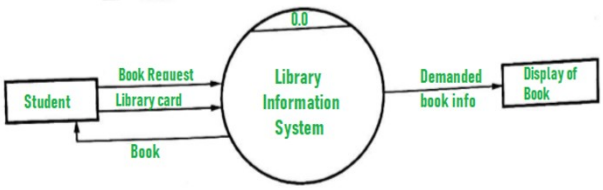
**2.Informal Methods:**

1.Brainstorming
2.Sketching and Diagramming
3.Storytelling
4.Interviews and Workshops
5.Prototyping
6.Use Cases and User Stories

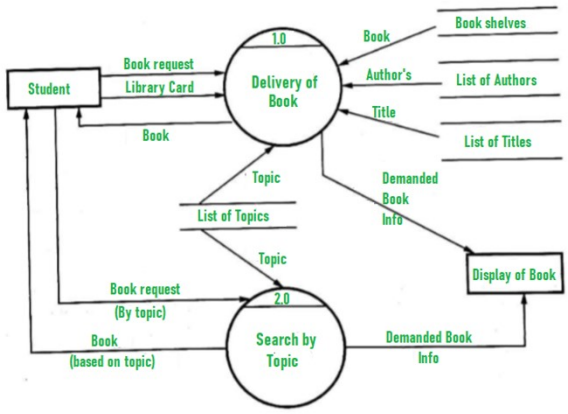
## Data Diagram Flow (DFD)

- DFDs visually depict data flow within a system using standardized symbols and notation.
- They illustrate how data moves between processes, data stores, and external entities.
- DFDs consist of processes (activities), data flows, data stores, and external entities.
- Standardized symbols like rectangles, arrows, circles, and squares represent components.
- They provide a visual overview of the system's data flow, aiding in understanding its structure, behavior, and interactions.

### 0-level DFD

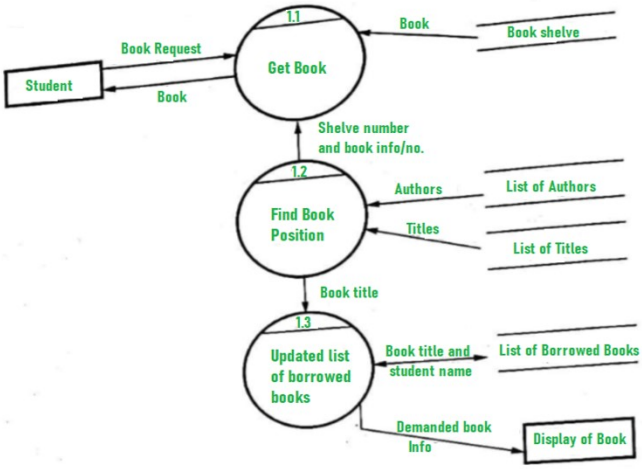



### 1-level DFD




Level 1 DFD

### 2-level DFD




Level 2 DFD

**General guidelines and rules for constructing DFDs :**

1. Choose meaningful names for processes and external entities.
2. Number the processes.
3. Avoid complex DFD.
4. Remember that a DFD is not a flow chart.
5. All names should be unique.
6. Make sure the DFD is internally consistent.
7. Processes are always running, they do not start or stop.
8. All data flows are named.
9. Numbers of processes at each level should be small (not more than 7).
10. A process numbered in a DFD is leveled into *n* processes then each new process is numbered *p*.1, *p*.2,.....*p.n*, respectively.
11. All data flows entering or outgoing from context DFD process should also be maintained in its leveled DFD.
12. It should not contain loops.
13. It should not contain crossing lines.

## Flowchart

- Graphical representation of process or algorithm.
  Shows sequence of steps or actions involved.
- Symbols represent actions, decisions, inputs, outputs.
  Visualizes logic and flow of system or program.
- Used for planning, designing, documenting, communicating workflows.
- **Flowcharting techniques aid software development by:**

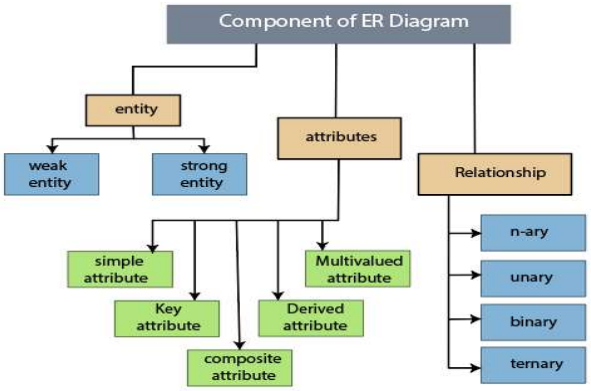
1.Visualizing Logic
2.Documentation
3.Analysis
4.Planning and Design
5.Debugging
6.Communication

## Differences b/w DFD and Flowchart

| Factors | DFD | Flowchart |
|---|---|---|
| **Focus** | Emphasize data flow within a system. | Detail process flow and logic. |
| **Level of Detail** | Provide a high-level overview | Offer detailed step-by-step depiction. |
| **Symbols** | Use standardized symbols for processes, data flows, etc. | Include additional shapes like diamonds for decisions. |
| **Purpose** | Analyze data flow in system design. | Model processes and algorithms. |

## ER model

- ER model provides a conceptual framework for database structure.
- It represents entities, such as objects or concepts, within the system.
- Illustrates the relationships between entities.
- Complex data structures are simplified into easily understandable components.
- Enhances clarity and consistency in data organization and storage.
- Serves as a foundational tool for database designers and developers.
- Guides in creating robust, scalable, and well-structured databases.
- Ensures accurate representation of data requirements within the system.




- **Advantages**:
  1.Clarity 3.Simplicity
  2.Ease of Communication 4.Normalization
- **Disadvantages**:
  1.Abstraction 3.Learning Curve
  2.Implementation Overhead 4.Complexity Handling

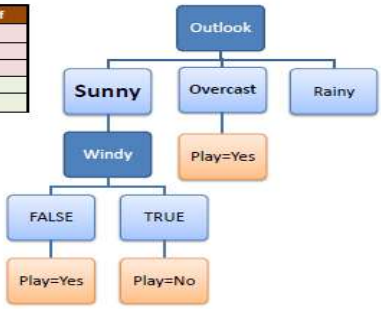
## Decision Tree:

- Graphical representation.
- Hierarchical structure.
- Decision rules shown as paths.
- Intuitive visualization.
- Handles categorical and numerical data.

## Decision Table:

- Tabular representation.
- Organized by conditions and decisions.
- Compact format.
- Systematic analysis.
- Potential readability issues with complexity.

| Temp | Humidity | Windy | Play Golf |
|---|---|---|---|
| Mild | High | FALSE | Yes |
| Cool | Normal | FALSE | Yes |
| Mild | Normal | FALSE | Yes |
| Cool | Normal | TRUE | No |
| Mild | High | TRUE | No |




## srs document

- srs document is a tool for communication b/w stack holders and s/w designers .
- Describes project, software, or application nature.
- Provides detailed overview, parameters, and goals.
- Defines target audience, user interface, hardware, and software requirements.

## IEEE standard for srs document

- **Introduction**: Overview of the software product and its purpose.
- **General Description:** Description of the project, target audience, and user interface.
- **Functional Requirements**: Details of the system's functionality and behavior.
- **Interface Requirements:** Specifications for interfaces with external systems or users.
- **Performance Requirements**: Performance criteria and expectations for the software.
- **Design Constraints:** Limitations or constraints on the design or implementation.
- **Non-Functional Attributes:** Characteristics such as reliability, usability, and security.
- **Preliminary Schedule and Budget:** Estimated timeline and budget for the project.
- **Appendices:** Additional supporting documentation, references, or resources

## Goal of srs document

1.Define requirements 4.Establish scope
2.Communicate expectations 5.Guide development
3.Facilitate agreement 6.Ensure quality

## Characteristics of SRS Document

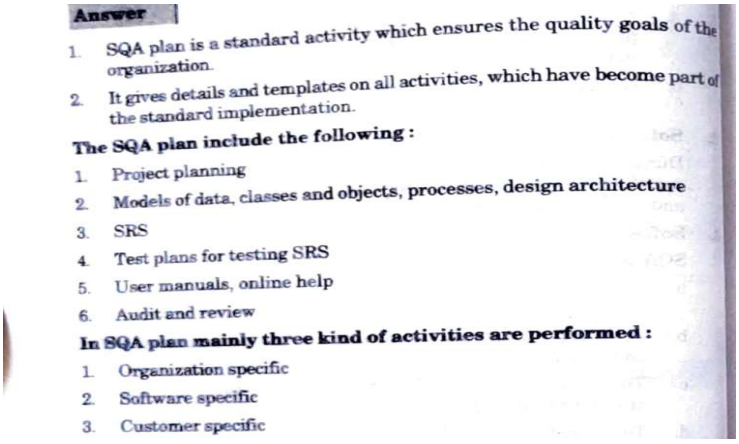
1.Comprehensive 6.Clear and concise
2.Specific 7.Measurable
3.Traceable 8.Consistent
4.Accurate 9.Verifiable
5.Stakeholder-oriented 10Dynamic

- **Advantages:**
  - Clarity in project objectives and requirements.
  - Alignment of stakeholders' understanding.
  - Guidance for development process.
  - Facilitation of stakeholder consensus.
  - Support for high-quality product development.
- **Disadvantages:**
  - Potential rigidity in adapting to changes.
  - Time-consuming to create and maintain.
  - Complexity may lead to confusion.
  - Risk of scope creep.
  - Ongoing maintenance required for relevance.

## Software Quality Assurance (SQA)

- Software Quality Assurance (SQA) refers to a systematic process that ensures the development and implementation of software products meet predefined quality standards and user requirements.
- **Here are the key points:**
  - **Goal**: Ensure software meets quality standards and user requirements.
  - **Process-oriented:** Follow systematic processes throughout development.
  - **Preventive**: Focus on defect prevention early in the process.
- **Standards Compliance:** Adhere to industry standards and regulations.
- **Continuous Improvement**: Promote ongoing refinement of processes.
  - **Testing**: Employ various testing techniques for validation.
  - **Documentation**: Maintain comprehensive documentation for transparency.
  - **Collaboration**: Involve stakeholders in identifying and resolving issues.
  - **Metrics**: Use metrics to assess software quality and process performance.
  - **Customer Satisfaction:** Aim to deliver software that satisfies user needs.

## SQA Plans



**Answer**

1. SQA plan is a standard activity which ensures the quality goals of the organization.
2. It gives details and templates on all activities, which have become part of the standard implementation.

**The SQA plan include the following :**

1. Project planning
2. Models of data, classes and objects, processes, design architecture
3. SRS
4. Test plans for testing SRS
5. User manuals, online help
6. Audit and review

**In SQA plan mainly three kind of activities are performed :**

1. Organization specific
2. Software specific
3. Customer specific

## ISO 9000 series

- The ISO 9000 series of standards is a set of guidelines established by the International Standards Organization (ISO) to promote standardization and quality management practices across industries.
- **ISO 9001:** For design, development, production, and servicing, often used by software development firms.
- **ISO 9002:** For production-only organizations like steel and car manufacturing, less relevant to software.
- **ISO 9003**: For installation and testing, such as gas companies, less applicable to software development.

- **Benefits**:
  - Improved quality and customer satisfaction.
  - Enhanced efficiency and competitiveness.
  - Compliance with regulatory requirements.
  - Increased credibility and market opportunities.
- **Limitations**:
  - Time, resource, and cost-intensive.
  - Reduced flexibility and innovation.
  - Emphasis on documentation may lead to bureaucracy.
  - High certification costs.

## SEI Capability Maturity Model (CMM)

- The SEI Capability Maturity Model (CMM) helps organizations improve their software development processes by providing a structured path from chaotic to mature processes.
- **The CMM comprises five levels of process maturity:**
  - **Initial**: Processes are ad hoc and unpredictable.
  - **Repeatable**: Basic project management processes are established.
  - **Defined**: Processes are documented and standardized.
  - **Managed**: Detailed measures of process performance are collected.
  - **Optimizing**: Continuous process improvement is institutionalized.

## Verification:

- Evaluates software against specified requirements and standards.
- Includes reviews, inspections, and walkthroughs.
- Ensures correct implementation according to design.
- Checks adherence to defined specifications at each stage.

## Validation:

- Evaluates software to meet customer needs.
- Involves testing with real users or stakeholders.
- Confirms fulfillment of intended purpose in real-world.
- Verifies solving the right problems and delivering expected benefits.