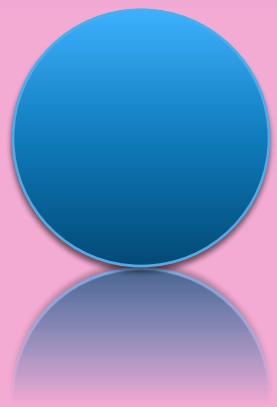


Module 3:
Requirements Analysis with Cost Estimation

Avinash Shrivas

Functional Requirements

- **Functional Requirement (What the system should do)**
- Functional requirements define the core features and functions of a system. They describe how the system behaves in different scenarios and what services it provides.
- **Key Characteristics of Functional Requirements:**

✓ Define what the system should do

✓ Describe input, process, and output

✓ Specify user interactions and system operations

✓ Can be tested and validated

Functional Requirements contd..

Example: Online Shopping Website

- A user should be able to **register** and **log in**.
- The system should allow adding/removing items from the cart.
- The website should send **order confirmation emails**.

Example 2: ATM machine

- It should allow users to **insert their card and enter a PIN**.
- It should provide options like **withdraw cash, check balance, and deposit money**.

Example 3: Hospital Management System

- A hospital receptionist should be able to register new patients.
- Doctors should be able to view patient medical history.
- The system should generate prescriptions and medical reports.
- Patients should be able to book appointments online.

Non Functional Requirements

Non-Functional Requirement (How the system should perform)

- Non-functional requirements focus on the **quality** of the system, like speed, security, and user experience.
- These are not related to specific system functionalities but describe how well the system should perform under various conditions.

Key Characteristics of Non-Functional Requirements

✅ Define **how** the system should behave

✅ Focus on **performance, security, usability, scalability, and reliability**

✅ Affect the **user experience and efficiency**

✅ Difficult to test but crucial for system success

Types & Examples of Non Functional Requirements

Performance Requirements

- The website should **load within 2 seconds** under normal conditions.
- The ATM should process a transaction **within 5 seconds**.
- The online shopping website should support **10,000 concurrent users** without crashing.

Security Requirements

- User data must be **encrypted using AES-256** encryption.
- Users should be required to **use two-factor authentication** for login.
- The system should **lock an account after five failed login attempts**.

Usability Requirements

- The mobile app should have an **intuitive user interface** for easy navigation.
- The ATM should support **multiple languages** for better user experience.
- The hospital management system should have **a simple dashboard for doctors** to access patient records easily.

Types & Examples of Non Functional Requirements *contd..*

Reliability & Availability Requirements

- The banking system should be available **99.99% of the time**.
- The hospital appointment booking system should work **24/7**.
- The e-commerce website should have **automatic failover** in case of server failure.

Maintainability & Scalability Requirements

- The software should be **easily upgradable** without downtime.
- The system should handle a **50% increase in traffic** without performance loss.
- The ERP system should be **modular** so that new features can be added easily.

Non Functional Requirements

Example 1: Online Shopping Website

- The website should **load within 3 seconds**.
- It should handle **10,000 users at a time** without crashing.
- User data must be **encrypted** for security.

Example 2: ATM machine

- The system should process transactions **within 5 seconds**.
- It should be available **24/7**.
- It should lock the user account after **3 incorrect PIN attempts** (security).

Comparison: Functional vs. Non-Functional Requirements

Feature	Functional Requirement	Non-Functional Requirement
Definition	Defines what the system should do	Defines how the system should perform
Focus	Features, logic, and operations	Performance, usability, security, and scalability
Examples (E-commerce)	User login, add to cart, checkout process	Page speed, security encryption, 24/7 uptime
Examples (Banking App)	Transfer money, check balance, pay bills	Transaction speed, fraud detection, high availability
Testing	Can be tested by running test cases	Hard to test, often requires performance benchmarking

SE GCR CODE

bsefg3s

Other Types of Requirements

Business Requirements (Why the system is needed)

- Business requirements define **high-level goals, objectives, and business needs** that the system should fulfill.
- **Example: Online Ticket Booking System**
- The system should **increase online ticket sales by 30%** in the next year.
- It should **reduce manual booking errors** by automating the process.

User Requirements (What users need)

- User requirements describe the **needs, expectations, and interactions** that users will have with the system. They are often represented as **user stories** or **use cases**.
- **Example: Ride-Sharing App (Uber, Ola)**
- A **driver** should be able to **accept or reject ride requests**.
- A **passenger** should be able to **track the driver's location**.

Other Types of Requirements contd..

System Requirements (Technical specifications)

- System requirements define the **hardware, software, and system configurations** required for the software to run.
- **Example: Hospital Management System**
- The system should run on **Windows Server 2019**.
- It should require **8GB RAM and 500GB storage**.

Regulatory & Compliance Requirements

- These requirements ensure that the software follows **legal, industry, and security standards**.
- **Example: Banking Software Compliance**
- Must follow **GDPR (General Data Protection Regulations) for user data privacy**.
- Must comply with **ISO 27001 for security standards**.

SRS

What is SRS?

- A **Software Requirements Specification (SRS)** is a formal document that defines the **functional and non-functional requirements** of a software system. It acts as a **blueprint** for developers, designers, testers, and stakeholders to ensure that the software meets business needs and user expectations.

Why is SRS Important?

✅ Ensures **clear communication** between stakeholders and developers

✅ Helps in **estimating costs, time, and resources**

✅ Reduces chances of **misunderstandings and scope creep**

✅ Acts as a **reference for testing and validation**

Structure of an SRS Document

An SRS document typically follows the IEEE 830 standard and consists of the following sections:

- **1. Introduction**
- **Purpose:** Why the software is being developed
- **Scope:** What the software will do and its benefits
- **Definitions, Acronyms, and Abbreviations:** Explaining key terms used in the document
- **References:** Any external documents, laws, or industry standards
- **Overview:** Brief outline of the rest of the document

- **2. Overall Description**
- **Product Perspective:** How this software fits into a larger system
- **Product Functions:** High-level summary of features
- **User Characteristics:** Who will use the system and their expertise level
- **Constraints:** Hardware, software, and regulatory limitations
- **Assumptions and Dependencies:** Conditions assumed to be true

Structure of an SRS Document

- **3.Specific Requirements**
- **Functional Requirements:** Features and functionalities of the software
- **Non-Functional Requirements:** Performance, security, usability, etc.
- **External Interface Requirements:** UI, APIs, hardware interactions
- **System Requirements:** Hardware, software, database, and network requirements

- **4. Appendices (Optional)**
- Additional supporting information like diagrams, tables, or references

Example of an SRS Document

Real-Life Example of an SRS Document: Online Food Ordering System (e.g., Swiggy, Zomato, UberEats)

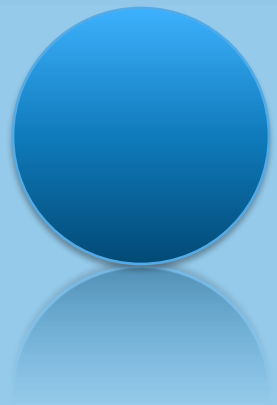
- **1. Introduction**
- **Purpose:** The system allows users to order food online from multiple restaurants.
- **Scope:**
- Customers can browse menus, place orders, and track delivery.
- Restaurants receive and prepare orders.
- Delivery personnel get assigned orders and deliver food.

- **2. Overall Description**
- **User Characteristics:**
- Customers: Basic tech knowledge, use mobile apps.
- Restaurants: Familiar with order management systems.
- Delivery Personnel: Uses mobile app for order tracking.
- **Constraints:**
- Must work on Android and iOS.
- Must handle **10,000+ concurrent users**.

Example of an SRS Document contd..

Real-Life Example of an SRS Document: Online Food Ordering System (e.g., Swiggy, Zomato, UberEats)

- **3. Specific Requirements**
- **Functional Requirements:**
- ✅ User should be able to **register, log in, and update profiles**.
  ✅ Customers should be able to **browse restaurants and filter food items**.
  ✅ Users can **add items to the cart and checkout**.
  ✅ Restaurants should be able to **accept, reject, and prepare orders**.
  ✅ Delivery personnel should receive order notifications and update delivery status.
- **Non-Functional Requirements:**
- The system should load within **3 seconds**.
- Payment transactions should be **secured with SSL encryption**.
- The mobile app should be **responsive on all screen sizes**.



Requirement Analysis and Requirement Engineering

1. Requirement Engineering

Requirement Engineering provides the appropriate mechanism for,

- Understanding what the customer wants,
- Analyzing need,
- Assessing feasibility,
- Negotiating a reasonable solution,
- Specifying the solution unambiguously,
- Validating the specification,
- Managing the Requirements as they are transformed into an operational system.

Requirements Engineering Tasks:

The software requirements engineering process includes the following steps of activities:

1. **Inception**
2. **Elicitation**
3. **Elaboration**
4. **Negotiation**
5. **Specification**
6. **Validation**
7. **Requirements Management**

Requirements Engineering Tasks:

The software requirements engineering process includes the following steps of activities:

1. **Inception**
2. **Elicitation**
3. **Elaboration**
4. **Negotiation**
5. **Specification**
6. **Validation**
7. **Requirements Management**

1. Inception

- A basic understanding of the problem is gained and the nature of the solution is addressed.
- The following criteria have to be addressed by the software engineers:

- Understanding of the problem.
- The people who want a solution.
- Nature of the solution.
- Communication and collaboration between the customer and developer.

2. Elicitation

- gathering the requirements from the stakeholders
- The right people must be involved in this phase. The following problems can occur in the elicitation phase:
- **Problem of Scope:** The requirements given are of unnecessary detail, ill-defined, or not possible to implement.
- **Problem of Understanding:** Not having a clear-cut understanding between the developer and customer when putting out the requirements needed. Sometimes the customer might not know what they want or the developer might misunderstand one requirement for another.
- **Problem of Volatility:** Requirements changing over time can cause difficulty in leading a project. It can lead to loss and wastage of resources and time.

1.2 Requirement Elicitation and Analysis

Requirement Elicitation and Analysis include following activity.

- **Domain understanding.**
- **Requirements collection.**
- **Classification.**
- **Conflict resolution.**
- **Define Priority.**
- **Requirements Checking.**

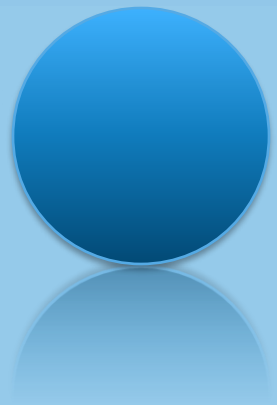
Requirement Elicitation and Analysis

- There are various ways to discover requirements
  - Interviews: open, closed, written,oral,one to one, group etc…
  - Surveys
  - Questionnaires
  - Task analysis
  - Domain Analysis
  - Brainstorming
  - Prototyping
  - Observation

3. Elaboration

- It takes the requirements that have been stated and gathered in the first two phases and refines them. Expansion and looking into it further are done as well.
- The main task in this phase is to indulge in modeling activities and develop a prototype that elaborates on the features and constraints using the necessary tools and functions.

System Modeling.

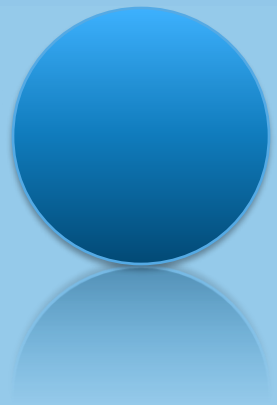


A model provides the blueprints of a system. Models may encompass detailed plans, as well as more general plans that give a 30,000-foot view of the system under consideration

Through modeling, we achieve four aims:

1. Models help us to visualize a system as it is or as we want it to be.
2. Models permit us to specify the structure or behavior of a system.
3. Models give us a template that guides us in constructing a system.
4. Models document the decisions we have made

System Modeling.

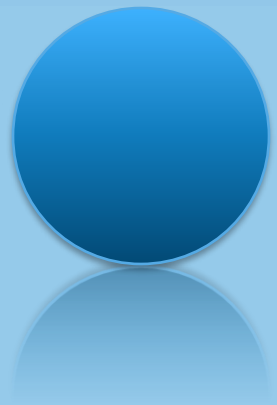


- Context Models. (e.g. Process model, Context model)
- Behavioral Models. (e. g. DFD, State machine model)
- Data Models. (E-R diagram, Data Dictionary)
- Object Models. (e.g. Class diagram, Object diagram)

4. Negotiation

- Discussion and exchanging conversation on what is needed and what is to be eliminated.
- Negotiation is between the developer and the customer and they dwell on how to go about the project with limited business resources.
- The following are discussed in the negotiation phase:
  - Availability of Resources.
  - Delivery Time.
  - Scope of requirements.
  - Project Cost.
  - Estimations on development.

5. Specification (SRS)



- This is the fifth phase of the requirements analysis process. This phase specifies the following:
- Written document.
- A set of models.
- A collection of use cases.
- A prototype.

6. Requirement Validation.

During the requirements validation process, following types of checks should be carried out on the requirements in the requirements documents.

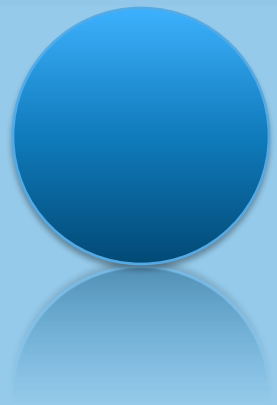
Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

1. Validity checks.
2. Consistency checks.
3. Completeness checks.
4. Realism checks.
5. Verifiability.

Requirements checking

- Validity. **Does the system provide the functions which best support the customer's needs?**
- Consistency. **Are there any requirements conflicts?**
- Completeness. **Are all functions required by the customer included?**
- Realism. **Can the requirements be implemented given available budget and technology**
- Verifiability. **Can the requirements be checked?**

Requirement Validation. (cont)



Requirements Validation Techniques:

1. Requirements reviews.
2. Prototyping.
3. Test Case generation.
4. Automated Consistency analysis.

7. Requirement Management

Requirement Management Plan is based on:

1. **Requirements identification.**

   - Each Requirement is assigned a unique identifier.

   <Requirement Type> <requirement #>

   Type: F : Functional Requirement.

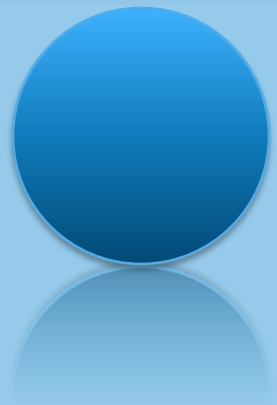
   D : Data Requirement.

   B : Behavioral Requirement.

   I :Interface Requirement.

   P : Output Requirement.

7. Requirement Management (Cont..)



2. A change management process.

3. Traceability policies.

   - Feature Traceability.

   - Source

   - Dependency

   - Subsystem

   - Interface

4. CASE tools support.

Feasibility studies

- A feasibility study decides whether or not the proposed system is worthwhile.
- A short focused study that checks
  - If the system contributes to organisational objectives;
  - If the system can be engineered using current technology and within budget;
  - If the system can be integrated with other systems that are already used.

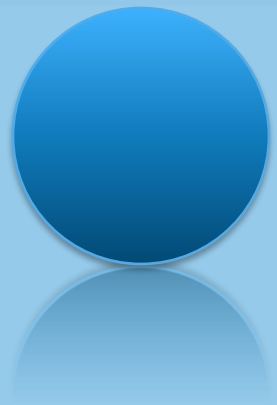
Feasibility study implementation

- Based on information assessment (what is required), information collection and report writing.
- Questions for people in the organisation
  - What if the system wasn't implemented?
  - What are current process problems?
  - How will the proposed system help?
  - What will be the integration problems?
  - Is new technology needed? What skills?
  - What facilities must be supported by the proposed system?

Changing requirements

- **The business and technical environment of the system always changes after installation.**
  - **New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by.**
- **The people who pay for a system and the users of that system are rarely the same people.**
  - **System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.**

Changing requirements



- **Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.**
  - **The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed.**