

# Software Engineering (CSE3004)

## Requirement Analysis and Specification



**Puneet Kumar Jain**

CSE Department

**National Institute of Technology Rourkela**

# Reference



- Rajib Mall, Introduction to Software Engineering
- Reference to his video lecture:  
<https://www.youtube.com/watch?v=l9XFipXoJb0&list=PLbRMhDVUMngf8oZR3DpKMvYhZKga90JVt&index=15>



# What are Requirements?

- A Requirement is:

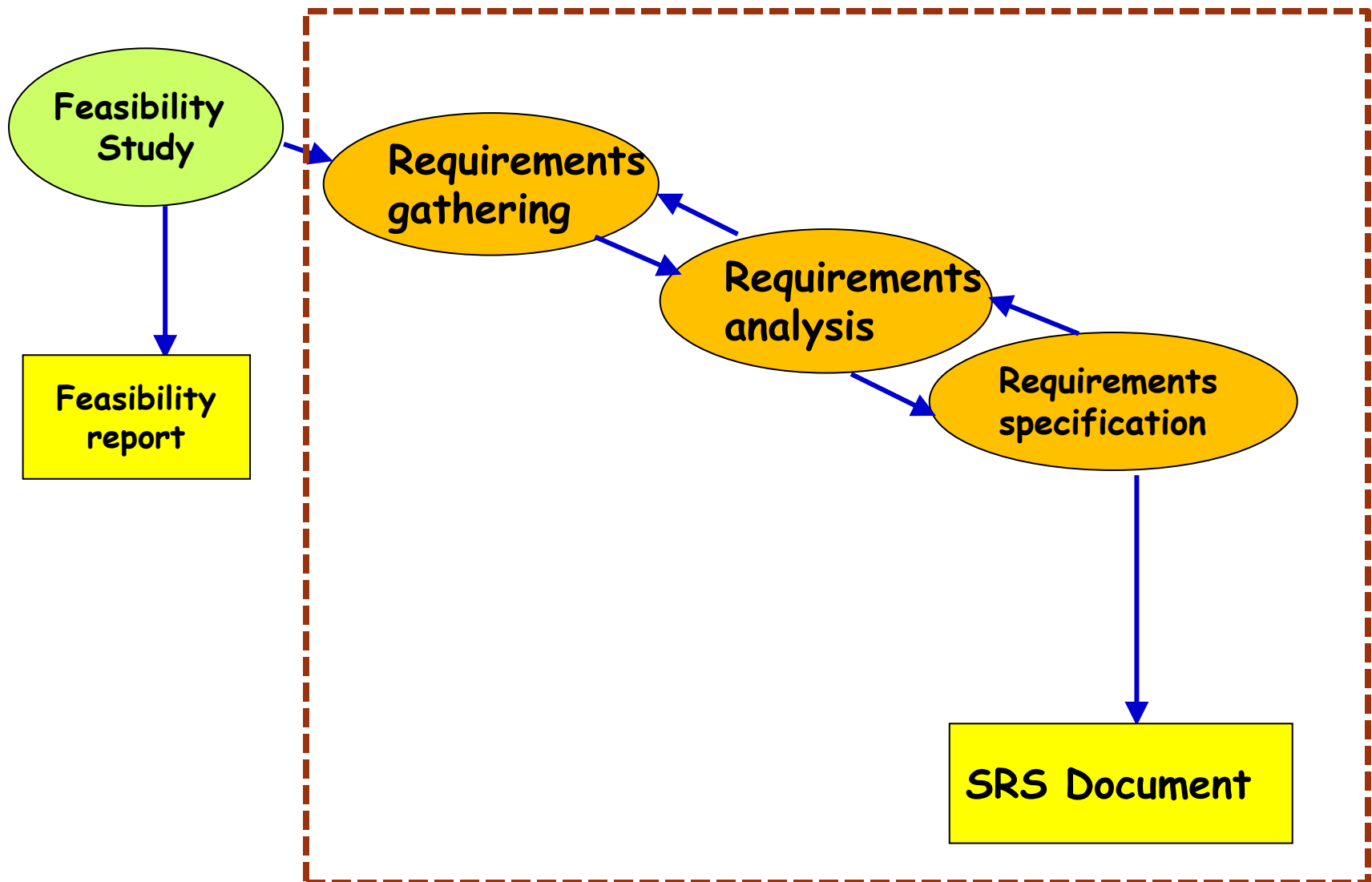
- A capability or condition required from the system.

- What is involved in requirements analysis and specification?
  - Determine what is expected by the client from the system.  
(Gather and Analyze)
  - Document those in a form that is clear to the client as well as to the development team members. (Document)

# Understanding and specifying requirements

- **For toy problems:** understanding and specifying requirements is rather easy...
- **For industry-standard problems:** Probably the hardest, most problematic and error prone among development tasks...
- The task of requirements specification :
  - **Input:** User needs that are hopefully fully understood by the users.
  - **Output:** Precise statement of what the software will do.

# Requirements Engineering Process



# Requirements Analysis and Specification



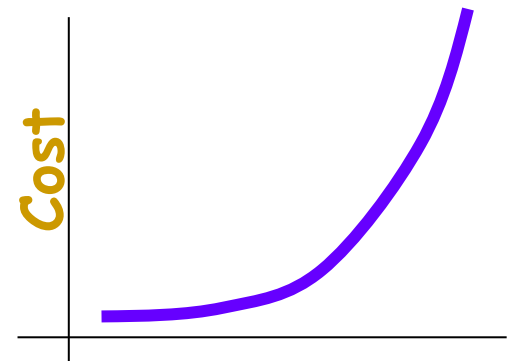
- Requirements Gathering:
  - Fully understand the user requirements.
  
- Requirements Analysis:
  - Remove inconsistencies, anomalies, etc. from requirements.
  
- Requirements Specification:
  - Document requirements properly in an SRS document.

# Need for SRS...

- **Good SRS reduces development cost:**
  - Requirement errors are expensive to fix later
  - Requirement changes cost a lot (typically 40% of requirements change later)
  - Good SRS can minimize changes and errors
  - **Substantial savings --- effort spent during requirement saves multiple times that effort**

- **An Example:**

Cost of fixing errors in requirement, design, coding, acceptance testing and operation increases exponentially





# What are the Uses of an SRS Document?











- Establishes the basis for agreement between the customers and the suppliers
- Forms the starting point for development.
- Provide a basis for estimating costs and schedules.
- Provide a basis for validation and verification.
- Provide a basis for user manual preparation.
- Serves as a basis for later enhancements.

# Forms A Basis for User Manual

- The SRS serves as the basis for writing User Manual for the software:
- **User Manual: Describes the functionality from the perspective of a user --- An important document for users.**
- Typically also describes how to carry out the required tasks with examples.

## <User Manual Template >

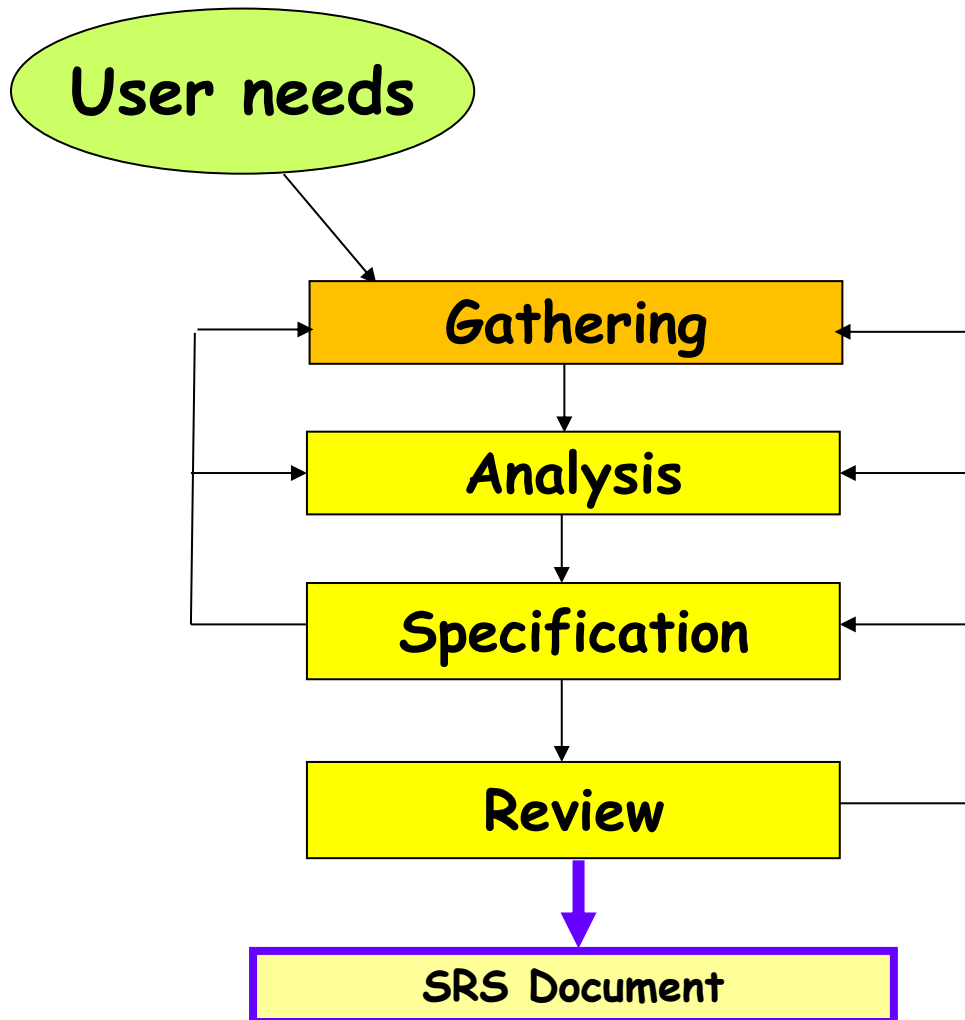
Throughout this document, the pictograms below are used to underline points or important notions

	Important information
	Good to know - Tricks
	<b>Risk</b> in front of a parameter setting or of a specific action
	Action to be avoided
	Mandatory action
	Sensitive or difficult procedure. To take into account necessarily
	Actions reserved for the .... < Adapt the vocabulary according to the application (administrator, manager, etc. >
	Actions reserved for the ... < Adapt the vocabulary according to the application (user, publisher, etc. >

# SRS Document: Stakeholders

- SRS intended for a diverse audience:
  - Customers and users use it for validation, contract, ...
  - Systems (requirements) analysts
  - Developers, programmers to implement the system
  - Testers use it to check whether requirements have been met
  - Project Managers to measure and control the project
- Different levels of detail and formality is needed for each audience
- Different templates for requirements specifications used by companies:
  - Often variations of [IEEE 830](#)

# Requirement process..



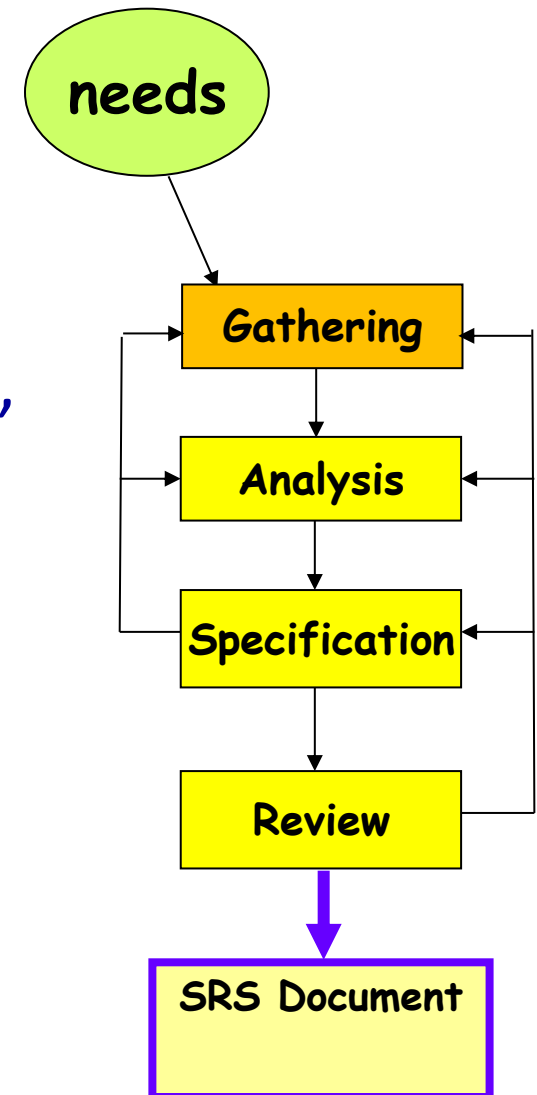
- Specification and review may lead to further gathering and analysis.

# Problems of requirements elicitation

- Stakeholders don't know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organisational and political factors may influence the system requirements.
- The requirements change during the analysis process. New stakeholders may emerge and the business environment may change.

# How to Gather Requirements?

- Observe existing (manual) systems,
- Study existing procedures,
- Discuss with customer and end-users,
- Input and Output analysis
- Analyze what needs to be done



# Requirements gathering

Possible stages include:

- Requirements discovery
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- Requirements classification and organisation
  - Groups related requirements and organises them into coherent clusters.
- Prioritisation and negotiation
  - Prioritising requirements and resolving requirements conflicts.
- Requirements specification
  - Requirements are documented and input into the next round of the spiral.

Reference: Chapter 4 Requirements Engineering, Sommerbille

# Requirements Gathering Activities

1. Study existing documentation
2. Interview :
  - Formal or informal
3. Task analysis
  - Requirements to performs the desired task
4. Scenario analysis
  - Requirements considering various scenarios
5. Form analysis
  - Considering the forms use to receive the input and output



# Interviewing



- Formal or informal interviews with stakeholders are part of most RE processes.
- Types of interview
  - **Closed interviews** based on pre-determined list of questions
  - **Open interviews** where various issues are explored with stakeholders.
- Effective interviewing
  - Be open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to stakeholders.
  - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

# Scenarios

- A structured form of user story
  
- Scenarios should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

# Requirements Gathering (CONT.)

- In the absence of a working system,
  - Lot of imagination and creativity are required.
- Interacting with the customer to gather relevant data:
  - Requires a lot of experience.
- Some desirable attributes of a good requirements analyst:
  - Good interaction skills,
  - Imagination and creativity,
  - Experience...

## Case Study: Automation of Office Work at CSE Dept.

- The academic, inventory, and financial information at the CSE department:
  - At present carried through manual processing by two office clerks, a store keeper, and two attendants.
  
- Considering the low budget he had at his disposal:
  - The HoD entrusted the work to a team of student volunteers.

## Case Study: Automation of Office Work at CSE Dept.

- The team was first briefed by the HoD:
  - Concerning the specific activities to be automated.
- The analysts first discussed with the two office clerks:
  - Regarding their specific responsibilities (tasks) that were to be automated.
- The analyst also interviewed student and faculty representatives who would also use the software.

**Interview**

# Case Study: Automation of Office Work at CSE Dept.

- For each task that a user needs the software to perform, they asked:
  - The steps through which these are to be performed.
  - The various scenarios that might arise for each task.
- Also collected the different types of forms that were being used.

**Task and Scenario Analysis**

**Form Analysis**

# Case Study: Automation of Office Work at CSE Dept.

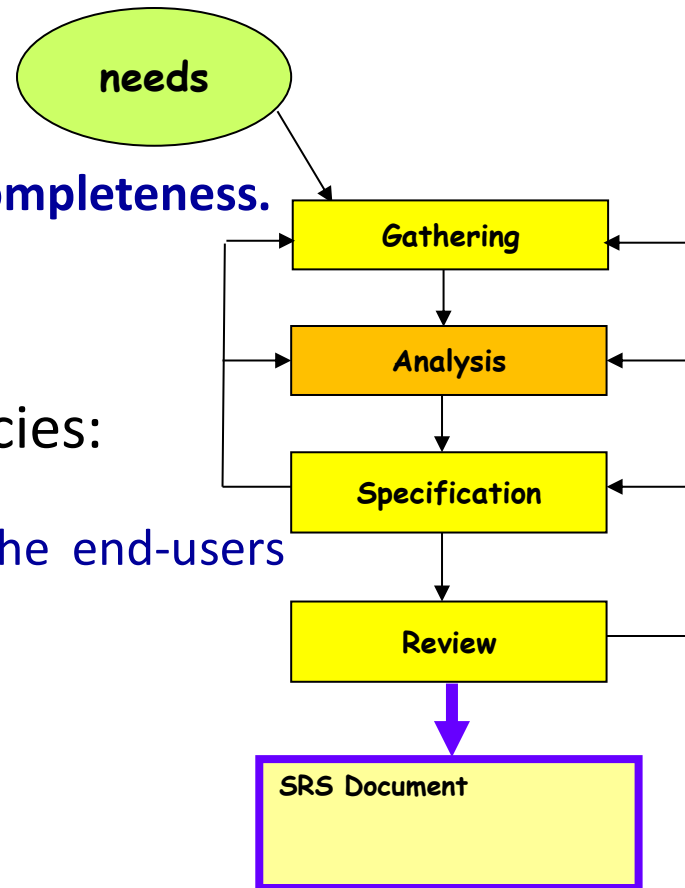
- The analysts understood the requirements for the system from various user groups:
  - Identified inconsistencies, ambiguities, incompleteness.
- Resolved the requirements problems through discussions with users:
  - Resolved a few issues which the users were unable to resolve through discussion with the HoD.
- Documented the requirements in the form of an SRS document.

**Requirements Analysis**

**Requirements Specification**

# Analysis of Gathered Requirements

- Main purpose of req. analysis:
  - Clearly understand user requirements,
  - **Detect inconsistencies, ambiguities, and incompleteness.**
- Remove incompleteness and inconsistencies:
  - Resolved through further discussions with the end-users and the customers.





**“When temperature becomes high, start cooler”**

**Do you notice any problems?**

- Above what threshold we consider the temperature to be high?

# Inconsistent Requirement



**Some part of the requirement contradicts some other requirement.**

- **Example:**

- One customer says turn off heater and open water shower when temperature  $> 100^{\circ}\text{C}$
- Another customer says turn off heater and turn ON cooler when temperature  $> 100^{\circ}\text{C}$

## Some requirements not included: Possibly due to oversight.

- Example:

- The analyst has not recorded that when temperature falls below 90°C :
  - heater should be turned ON
  - water shower turned OFF.

# Analysis of the Gathered Requirements

- Some anomalies and inconsistencies can be very subtle:
  - Escape even most experienced eyes.
  - If a formal specification of the system is constructed,
    - Many of the subtle anomalies and inconsistencies get detected.

# Analysis of the Gathered Requirements (CONT.)

- Experienced analysts take considerable time:
  - **Clearly understand the exact requirements the customer has in his mind.**
- Experienced systems analysts know - often as a result of painful experiences ---

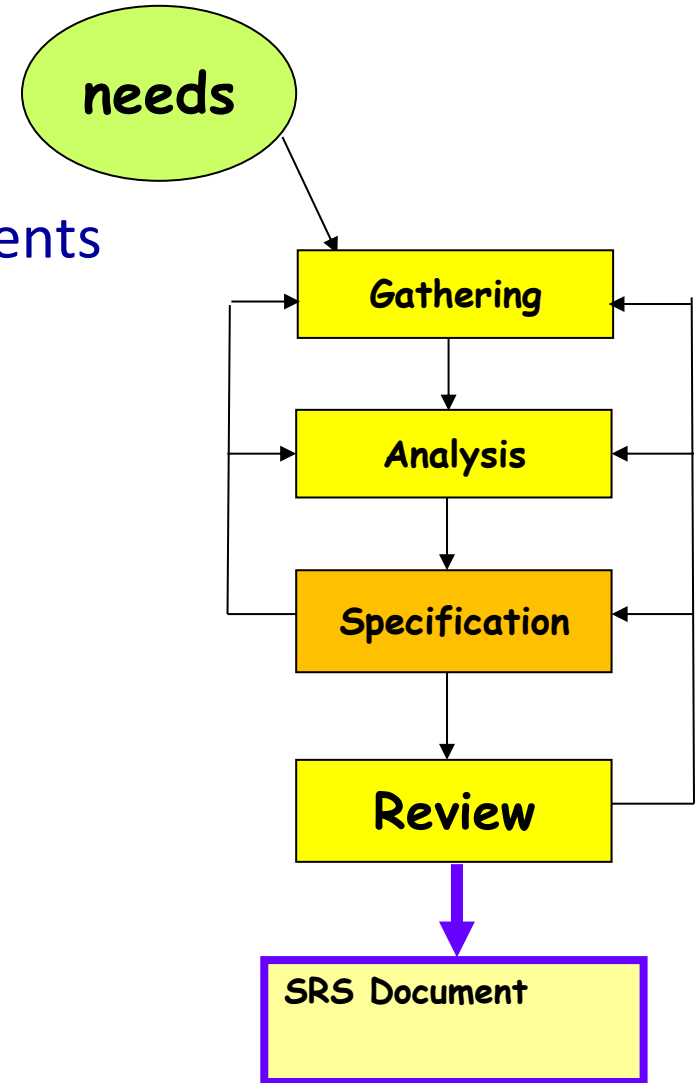
**“Without a clear understanding of the problem, it is impossible to develop a satisfactory system.”**

# Analysis of the Gathered Requirements(CONT.)

- After collecting all data regarding the system to be developed,
  - Remove all inconsistencies and anomalies from the requirements,
  - Systematically organize requirements into a **Software Requirements Specification (SRS)** document.

# Software Requirements Specification

- Main aim:
  - Systematically organize the requirements arrived during requirements analysis.
  - Document requirements properly.



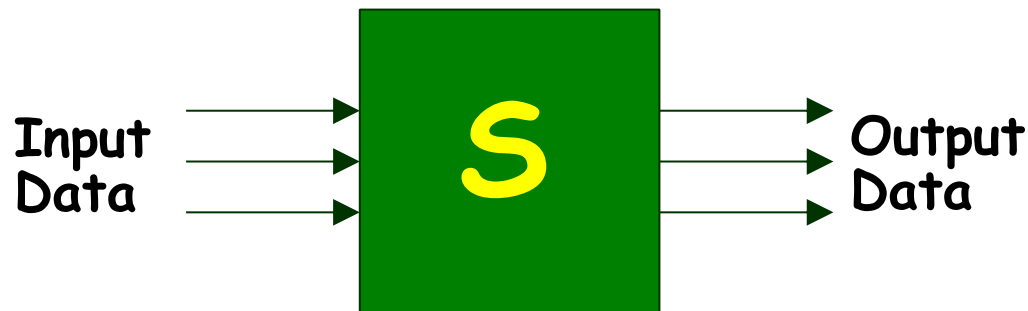
# SRS Document

- As already pointed out--- useful in various contexts:
  - **Statement of user needs**
  - **Contract document**
  - **Reference document**
  - **Definition for implementation**



# SRS Document (CONT.)

- SRS document is known as **black-box specification**:
  - The system is considered as a black box whose internal details are not known.



- Only its visible external (i.e. input/output) behavior is documented.

# SRS Document (CONT.)



- SRS document concentrates on:
  - What needs to be done in terms of input-output behaviour
  - Carefully avoids the solution (“how to do”) aspects.
- The requirements at this stage:
  - Written using end-user terminology.
- If necessary:
  - Later a formal requirement specification may be developed from it.

# Properties of a Good SRS Document

- **It should be concise**
  - and at the same time should not be ambiguous.
- **It should specify what the system must do**
  - and not say how to do it.
- **Easy to change.,**
  - i.e. it should be well-structured.
- **It should be consistent.**
- **It should be complete.**

# Properties of a Good SRS Document (cont...)



- **It should be traceable**

- You should be able to trace which part of the specification corresponds to which part of the design, code, etc and vice versa.

- **It should be verifiable**

- e.g. “system should be user friendly” is not verifiable

# SRS should not include...

- **Project development plans**

- E.g. cost, staffing, schedules, methods, tools, etc
  - **Lifetime of SRS is until the software is made obsolete**
  - **Lifetime of development plans is much shorter**

- **Product assurance plans**

- Configuration Management, Verification & Validation, test plans, Quality Assurance, etc

- **Designs**

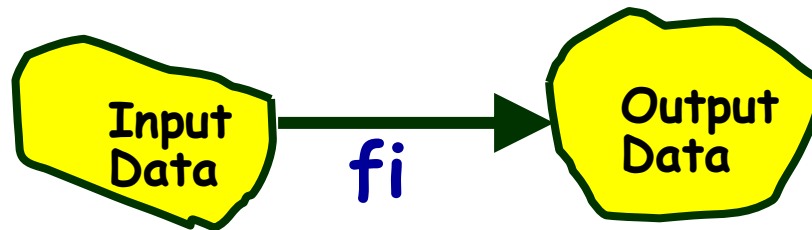
- Requirements and designs have different audiences
- Analysis and design are different areas of expertise

# SRS Document (CONT.)

- Four important parts:
  - **Functional requirements,**
  - **Non-functional requirements,**
  - **External Interfaces**
  - **Constraints**
  - **Goals of implementation**

# Functional Requirements

- Specifies all the functionality that the system should support
  - **Heart of the SRS document:**
  - **A set of high-level requirements**
- Outputs for the given inputs and the relationship between them



- Must specify behavior for invalid inputs too!

# Example: Functional Requirement

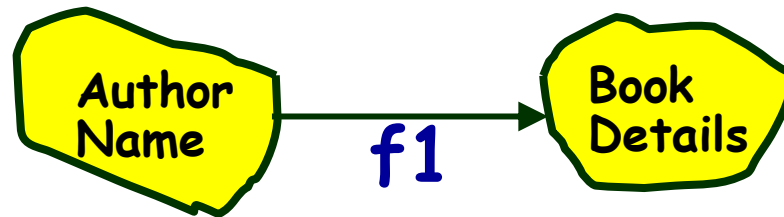
- F1: Search Book

- Input:

- an author's name:

- Output:

- details of the author's books and the locations of these books in the library.





# Functional Requirement Documentation



## ■ Overview

- describe purpose of the function and the approaches and techniques employed

## ■ Inputs and Outputs

- sources of inputs and destination of outputs
- quantities, units of measure, ranges of valid inputs and outputs
- Timing

## ■ Processing

- validation of input data
- exact sequence of operations
- responses to abnormal situations
- any methods (eg. equations, algorithms) to be used to transform inputs to outputs

# Example Functional Requirements

- Req. 1:
  - Once user selects the “search” option,
    - he is asked to enter the key words.
  - The system should output details of all books
    - whose title or author name matches any of the key words entered.
    - Details include: Title, Author Name, Publisher name, Year of Publication, ISBN Number, Catalog Number, Location in the Library.

# Example Functional Requirements

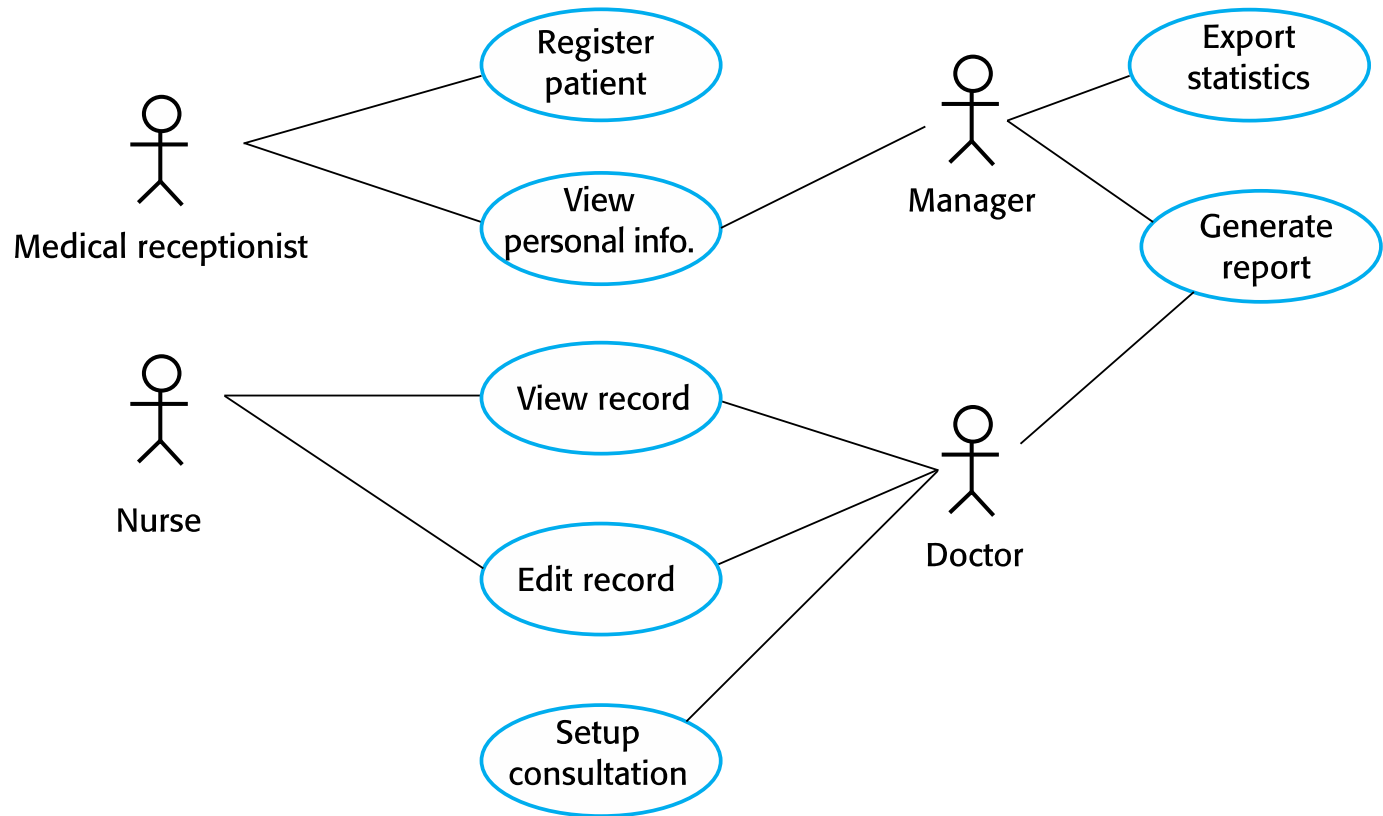
- Req. 2:
  - When the “renew” option is selected,
    - The user is asked to enter his membership number and password.
  - After password validation,
    - The list of the books borrowed by him are displayed.
  - The user can renew any of the books:
    - By clicking in the corresponding renew box.

# Use Cases



- A use case is a term in UML:
  - Represents a high level functional requirement.
- Use case representation is more well-defined and has agreed documentation:
  - Compared to a high-level functional requirement and its documentation
  - Therefore many organizations document the functional requirements in terms of use cases

# Use cases for the Mentcare system



# Non-functional Requirements

- Characteristics of the system which can not be expressed as functions:
  - Maintainability,
  - Portability,
  - Usability,
  - Security,
  - Safety, etc.

# Non-functional Requirements

- Reliability issues
- Performance issues:
  - **Example:** How fast can the system produce results?
    - At a rate that does not overload another system to which it supplies data, etc.
    - Response time should be less and **deterministic**
    - Needs to be measurable (verifiability)

# Constraints

- Hardware to be used,
- Operating system
  - or DBMS to be used
- Capabilities of I/O devices
- Standards compliance
- Data representations by the interfaced system



# External Interface Requirements

- User interfaces
- Hardware interfaces
- Software interfaces
- Communications interfaces with other systems
- File export formats

# Goals of Implementation

- Goals describe things that are desirable of the system:
  - But, would not be checked for compliance.
  - For example,
    - Reusability issues
    - Functionalities to be developed in future

# IEEE 830-1998 Standard for SRS

- Section 4 of IEEE 830 (How to produce a good SRS)
  - Nature of SRS
    - Functionality, interfaces, performance, qualities, design constraints
  - Environment of the SRS
    - Where does it fit in the overall project hierarchy
  - Characteristics of a good SRS
    - Generalization of the characteristics of good requirements to the document
  - Evolution of the SRS
    - Implies a change management process
  - Prototyping
    - Helps elicit software requirements and reach closure on the SRS
  - Including design and project requirement in the SRS
    - Focus on external behavior and the product, not the design and the production process (describe in a separate document)

# IEEE 830-1998 Standard for SRS

- Section 5 of IEEE 830
  - Contents of SRS
  - Introduction
  - General description of the software product
  - Specific requirements (detailed)
  - Additional information such as appendixes and index, if necessary

# IEEE 830-1998 Standard for SRS

- Title
- Table of Contents
- 1. Introduction
  - 1.1 Purpose
    - Describe purpose of the system
    - Describe intended audience
  - 1.2 Scope
    - What the system will and will not do
  - 1.3 Definitions, Acronyms, and Abbreviations
    - Define the vocabulary of the SRS (may also be in appendix)
  - 1.4 References
    - List all referenced documents and their sources SRS (may also be in appendix)
  - 1.5 Overview
    - Describe how the SRS is organized
- 2. Overall Description
- 3. **Specific Requirements**
- Appendices
- Index

# IEEE 830-1998 Standard – Section 2 of SRS



- Title
- Table of Contents
- 1. Introduction
- 2. **Overall Description**
  - 2.1 **Product Perspective**
  - 2.2 **Product Functions**
  - 2.3 **User Characteristics**
  - 2.4 **Constraints**
  - 2.5 **Assumptions and Dependencies**
- 3. Specific Requirements
- 4. Appendices
- 5. Index

•Present the business case and operational concept of the system  
•Describe external interfaces: system, user, hardware, software, communication  
•Describe constraints: memory, operational, site adaptation

•Summarize the major functional capabilities

•Describe technical skills of each user class

•Describe other constraints that will limit developer's options; e.g., regulatory policies; target platform, database, network, development standards requirements

# IEEE 830-1998 Standard – Section 3 of SRS (1)

- ...
- 1. Introduction
- 2. Overall Description
- 3. Specific Requirements
  - 3.1 External Interfaces
  - 3.2 Functions
  - 3.3 Performance Requirements
  - 3.4 Logical Database Requirements
  - 3.5 Design Constraints
  - 3.6 Software System Quality Attributes
  - 3.7 Object Oriented Models
- 4. Appendices
- 5. Index

Specify software requirements in sufficient detail so that designers can design the system and testers can verify whether requirements met.

State requirements that are externally perceivable by users, operators, or externally connected systems

Requirements should include, at the least, a description of every input (stimulus) into the system, every output (response) from the system, and all functions performed by the system in response to an input

# IEEE 830-1998 Standard – Templates

- Section 3 (Specific Requirements) can be organized in several different ways based on
  - **Modes** (expert mode, novice mode)
  - **User classes**
  - **Concepts (object/class)**
  - **Features**
  - **Stimuli**



# Example Section 3 of SRS of Academic Administration Software



- **SPECIFIC REQUIREMENTS**

- **3.1 Functional Requirements**

- **3.1.1 Subject Registration**

- The subject registration requirements are concerned with functions regarding subject registration which includes students selecting, adding, dropping, and changing a subject.

- **F-001:**

- The system shall allow a student to register a subject.

- **F-002:**

- It shall allow a student to drop a course.

- **F-003:**

- It shall support checking how many students have already registered for a course.

# Design Constraints (3.2)

## ■ 3.2 Design Constraints

### ■ C-001:

- AAS shall provide user interface through standard web browsers.

### ■ C-002:

- AAS shall use an open source RDBMS such as Postgres SQL.
- 

### ■ C-003:

- AAS shall be developed using the JAVA programming language

# Non-functional requirements

## ■ 3.3 Non-Functional Requirements

### ■ N-001:

- AAS shall respond to query in less than 5 seconds.

### ■ N-002:

- AAS shall operate with zero down time.

### ■ N-003:

- AAS shall allow upto 100 users to remotely connect to the system.

### ■ N-004:

- The system will be accompanied by a well-written user manual.

# Examples of Bad SRS Documents

- Unstructured Specifications:
  - Narrative essay --- one of the worst types of specification document:
    - Difficult to change,
    - Difficult to be precise,
    - Difficult to be unambiguous,
    - Scope for contradictions, etc.

# Examples of Bad SRS Documents

## ■ Noise:

- Presence of text containing information irrelevant to the problem.

## ■ Silence:

- Aspects important to proper solution of the problem are omitted.

# Examples of Bad SRS Documents

## ■ Overspecification:

- Addressing “how to” aspects
- For example, “Library member names should be stored in a sorted descending order”
- Overspecification restricts the solution space for the designer.

## ■ Contradictions:

- Contradictions might arise
  - if the same thing described at several places in different ways.

# Examples of Bad SRS Documents

## ■ Ambiguity:

- Literary expressions
- Unquantifiable aspects, e.g. “good user interface”

## ■ Forward References:

- References to aspects of problem
  - defined only later on in the text.

## ■ Wishful Thinking:

- Descriptions of aspects
  - for which realistic solutions will be hard to find.

# Suggestions for Writing Good Quality Requirements

- Keep sentences and paragraphs short.
- Use active voice.
- Use proper grammar, spelling, and punctuation.
- Use terms consistently and define them in a glossary.
- To see if a requirement statement is sufficiently well defined,
  - Read it from the developer's perspective



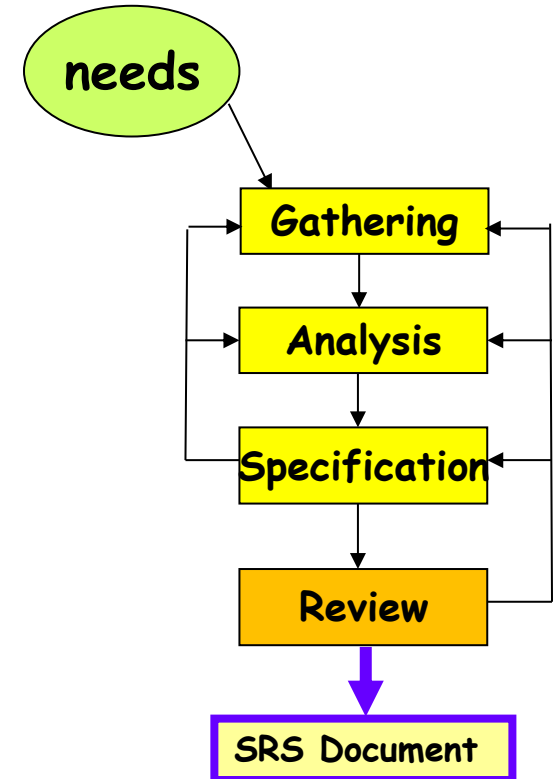
# Suggestions for Writing Good Quality Requirements



- Split a requirement into multiple sub-requirements:
  - Because each will require separate test cases and because each should be separately traceable.
  - If several requirements are strung together in a paragraph, it is easy to overlook one during construction or testing.

# SRS Review

- Review done by the Developers along with the user representatives.
- To verify that SRS confirms to the actual user requirements
- To detect defects early and correct them.
- Review typically done using standard inspection process:
  - Checklists.



# A Sample SRS Checklist

- Have response times been specified for functions ?
- Have all the HW, external SW and data interfaces been defined ?
- Is each requirement testable ?
- Is the initial state of the system defined ?
- Are the responses to exceptional conditions specified ?

# Summary



- Requirements analysis and specification
  - An important phase of software development:
  - Any error in this phase would affect all subsequent phases of development.
  
- Consists of two different activities:
  - Requirements gathering and analysis
  - Requirements specification

# Summary



- The aims of requirements analysis:
  - Gather all user requirements
  - Clearly understand exact user requirements
  - Remove inconsistencies and incompleteness.
  
- The goal of specification:
  - Systematically organize requirements
  - Document the requirements in an SRS document.

# Summary

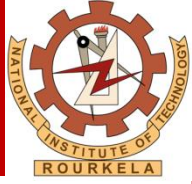
- Main components of SRS document:
  - Functional requirements
  - Non-functional requirements
  - Constraints
- Techniques to express complex logic:
  - Decision tree
  - Decision table

# Summary



- Formal requirements specifications have several advantages.
  - But the major shortcoming is that these are hard to use.

# Assignment



- A public library is considering the implementation of a computer-based system to help administer book loans at libraries.
- Identify the stakeholders in such a project.
- What might be the objectives of such a project and how might the success of the project be measured in practical terms?



# Stake holders

- Representatives of the local authority who own and pay for the library
- Library staff at various levels
- Technical support staff
- Software and hardware suppliers
- Library users

## ■ Objectives

- ‘To have in place an operational computer-based loans management system by dd/mm/yy which meets the requirements specified in Annex at a development cost of not more than e.g. Rs.10,00,000.’

## ■ Measures of effectiveness

- Was the application actually working by dd/mm/yy?
- Were costs under Rs. xxxx?
- Can users etc certify that delivered system meets the documented requirements?

# SRS Skeleton



## ■ 1.0 INTRODUCTION

- This document specifies the requirements for ...

## ■ 1.1 Purpose

- The purpose of the ...is to ....
- The system should assist ....
- The intended audience for this document is ...
- This specification describes .....

## ■ 1.2 Scope

- This document applies only to .....
- This specification is not concerned with .....

# Skeletal SRS

## ■ Introduction

- Purpose , the basic objective of the system
- Scope of what the system is to do , not to do

## ■ Overall description

- Product perspective
- Product functions
- User characteristics
- Assumptions
- Constraints

# Skeletal SRS...

- Specific requirements
  - External interfaces
  - Functional requirements
  - Performance requirements
  - Design constraints
  
- Acceptance criteria
  - desirable to specify this up front.

# Skeletal SRS



## 2.0 GENERAL DESCRIPTION

### 2.1 Product Perspective

- This system allows stakeholders to.....
- The system will display.....
- The system will help .....
- The system provides information about ....

### 2.2 Product Functions

The system provides the following functions:

# SRS Skeleton



- 2.3 User Characteristics
- The users of the system are:
  - Level of Users' Computer Knowledge
  - Level of Users' Business Knowledge
  - Frequency of Use
- 2.4 General Constraints
  - The system will support ....
  - The system will not allow .....
- 2.5 Assumption and Dependencies
  - This system relies on .....
  - The system must have a satisfactory interface and .....

- End of Chapter

*Thanks*