Requirement Analysis and Specification

Lecture#05-06

Dr. Sanjeev Patel

Asst. Professor,

Department of Computer Science and Engineering

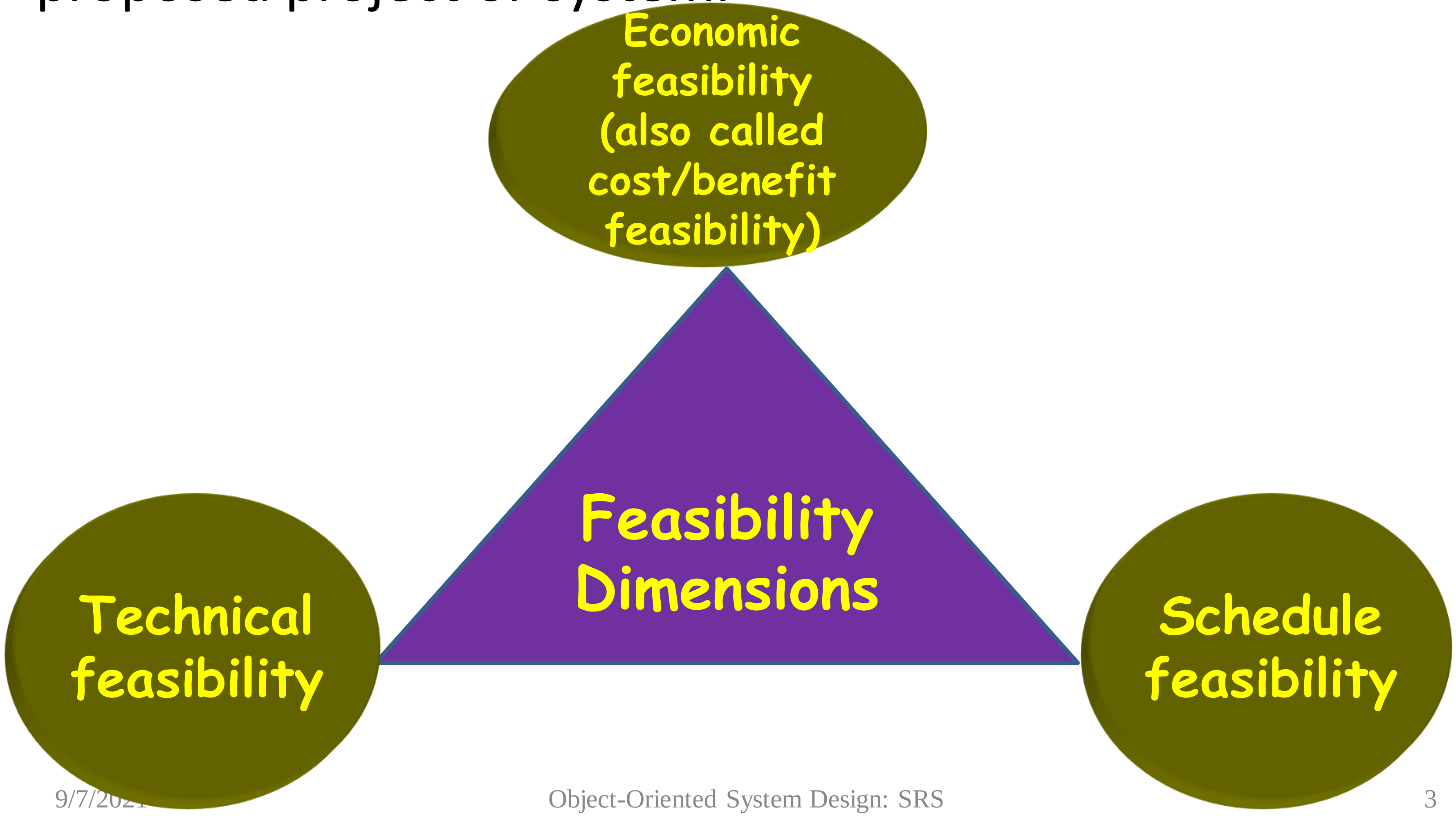
National Institute of Technology Rourkela, Odisha

Outline

- Feasibility Study and Analysis
- Requirements gathering
- Requirements analysis and determination
- Software requirements specification

Feasibility Study

- **Feasibility Study** is an assessment of the practicality of a proposed project or system.



Areas of Feasibility study

A. Technical feasibility: determine whether the company has the technical expertise to handle completion of the project.

- **Method of production**
  - Availability of inputs or raw materials and their quality and prices.
- **Production technique**
  - Tools and equipment needed for the project
- **Project location**
  - Availability of land (proper acreage and reasonable costs).

B. Financial feasibility

- In case of a new project, financial viability can be judged on the following parameters:
  - Total estimated cost of the project

Areas of Feasibility study

C. Schedule feasibility

- Feasibility is a measure of how reasonable the project timetable is?
- A project will fail if it takes too long to be completed before it is useful.
- It is necessary to determine whether the deadlines are mandatory or desirable.

D. Operational feasibility: It is the measure of

- how well a proposed system solves the problems?

E. Legal feasibility

- Determines whether the proposed system conflicts with legal requirements, e.g.,

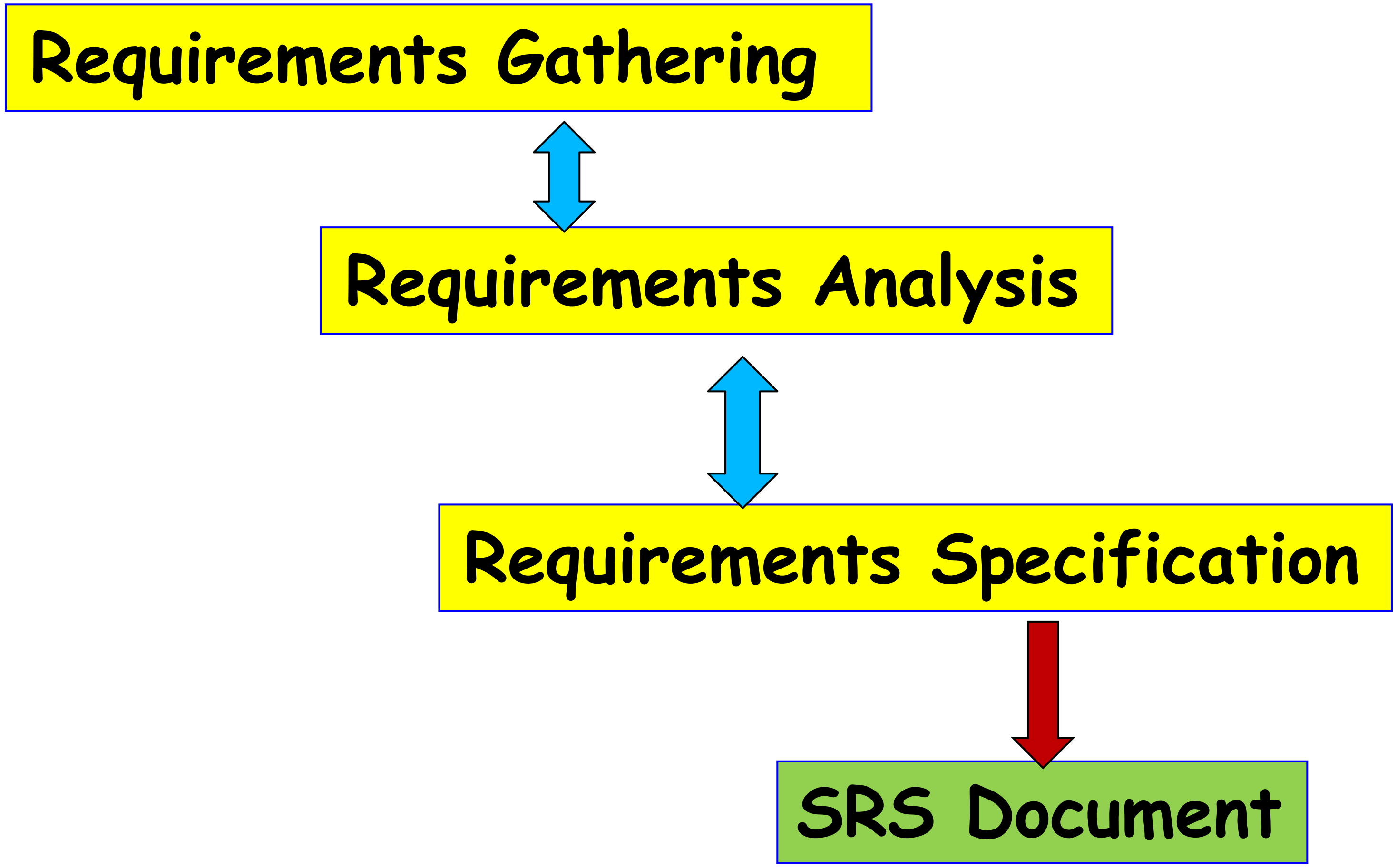
What are Requirements?

- 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)**

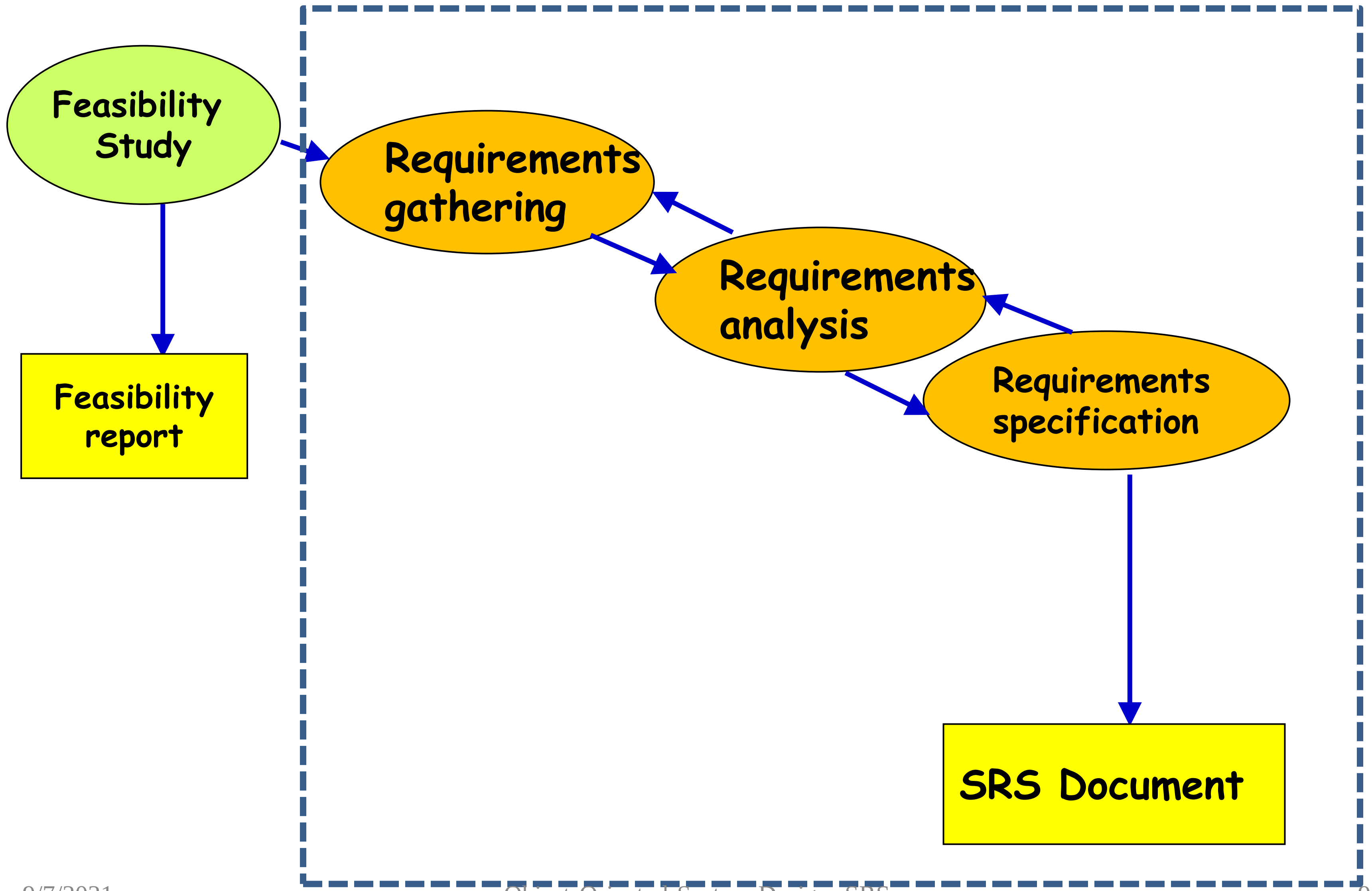
Understanding and specifying requirements

- **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.

Activities in Requirements Analysis and Specification



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.

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.

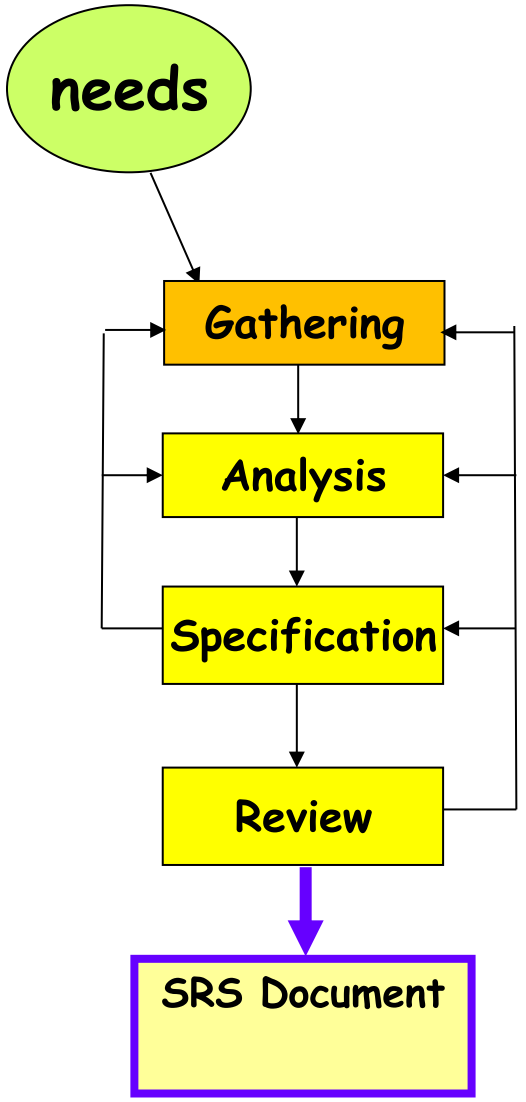
Problems of requirements elicitation

- 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.

Reference: Chapter 4 Requirements Engineering, Sommerbille

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.

Requirements Gathering Activities

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

Interviewing [3]

- 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 [3]

- 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 though 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.

Interview

- The analyst also interviewed student and faculty representatives who would also use the software.

Case Study: Automation of Office Work (CSE Dept.)

- For each task that a user needs the software to perform, they asked:

Task and Scenario Analysis

  - 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.

Form Analysis

Case Study: Automation of Office Work at CSE Dept.

- The analysts understood the requirements for the system from various user groups:

Requirements Analysis

  - Identified inconsistencies, ambiguities, incompleteness.

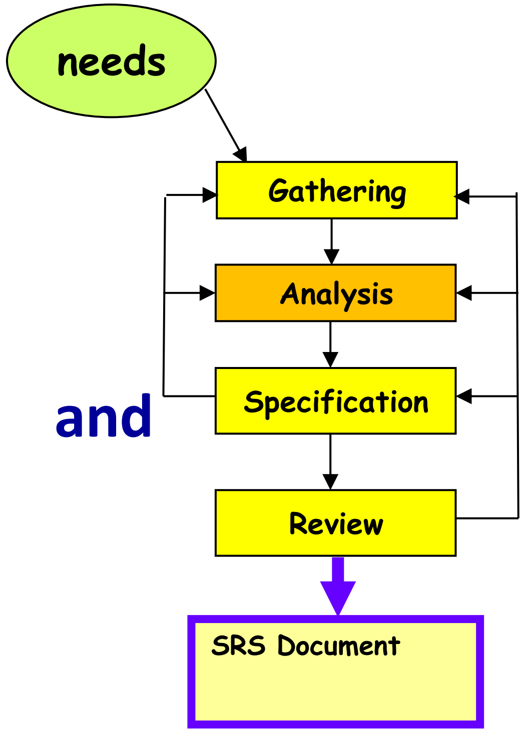
- Resolved the requirements problems through discussions with users:
  - Resolved a few issues which the users were unable to resolve through discussion with the HoD.

- Documented the requirements in the form of an SRS document.

Requirements Specification

Analysis of Gathered Requirements

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



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 ---

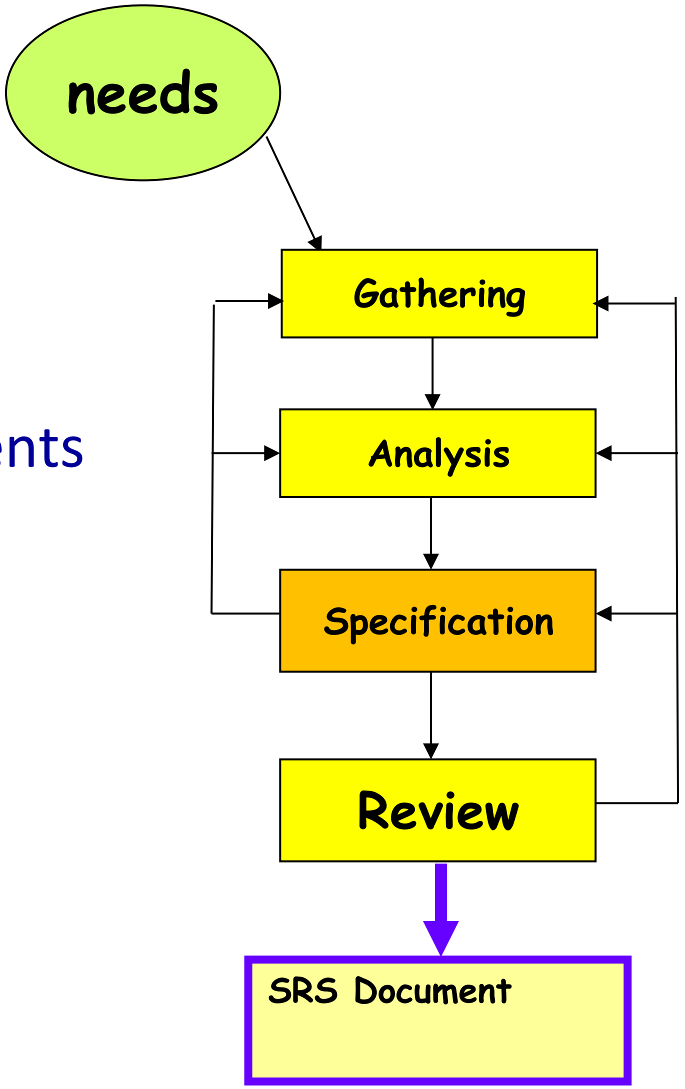
"Without a clear understanding of the problem, it is impossible to develop a satisfactory system."

Analysis of the Gathered Requirements(COND.)

- 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.

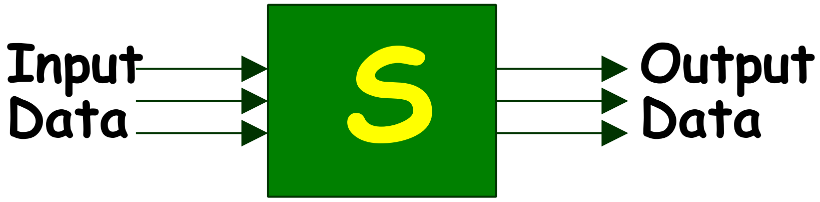


SRS Document

- 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) behaviour 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.

SRS Document (CONT.)

- 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...

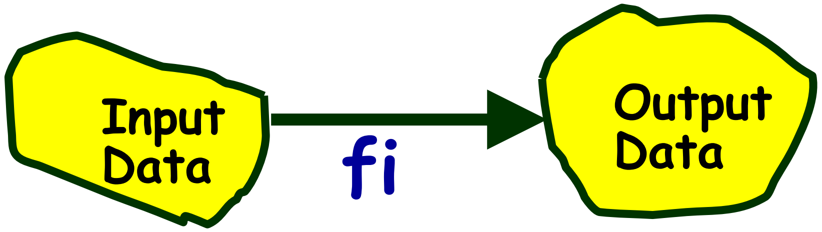
- **Project development plans**
  - E.g. cost, staffing, schedules, methods, tools, etc.
    - **Lifetime of SRS is until the software is made obsolete**
    - **Lifetime of development plans is much shorter**
- **Product assurance plans**
  - Configuration Management, Verification & Validation, test 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

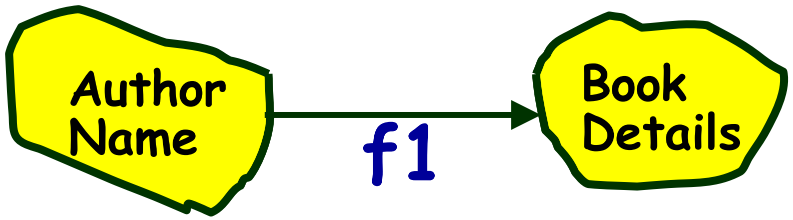
- Specifies all the functionality that the system should support
  - **Heart of the SRS document:**
  - **A set of high-level requirements**
- Outputs for the given inputs and the relationship between them



- Must specify behavior for invalid inputs too!

Example: Functional Requirements

- 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.

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

Interfaces

- 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

- Title
- Table of Contents
- 1. Introduction
  - 1.1 Purpose
  - 1.2 Scope
  - 1.3 Definitions. Acronyms, and Abbreviations
  - 1.4 References
  - 1.5 Overview
- 2. Overall Description
- 3. **Specific Requirements**
- Appendices
- Index

•Describe purpose of the system
•Describe intended audience

•What the system will and will not do

•Define the vocabulary of the SRS (may also be in appendix)

•List all referenced documents and their sources SRS (may also be in appendix)

•Describe how the SRS is organized

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 Requiremen
  - 3.1 External Interfaces
  - 3.2 Functions
  - 3.3 Performance Requirer
  - 3.4 Logical Database Requ
  - 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.

Example: 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.

Example: Bad SRS Documents

- **Noise:**
  - Presence of text containing information irrelevant to the problem.
- **Silence:**
  - Aspects important to proper solution of the problem are omitted.

Example: Bad SRS Documents

- **Overspecification:**
  - Addressing “how to” aspects
  - For example, “Library member names should be stored in a sorted descending order”
  - Over specification restricts the solution space for the designer.
- **Contradictions:**
  - Contradictions might arise
    - if the same thing described at several places in different ways.

Example: 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.

References

1. Rajib Mall, “Fundamentals of Software Engineering”, 3rd edition, PHI, 2009
2. R.S. Pressman, “Software Engineering: A Practitioner's Approach”, 7th Edition, McGraw
3. Sommerville, “ Introduction to Software Engineering”, 8th Edition, Addison-Wesley, 2007
4. JAMES RUMBAUGH, IVAR JACOBSON, GRADY BOOCH, “The Unified Modeling Language Reference Manual”, Second Edition, Addison-Wesley, 2004.
5. PPT available for the respective books

Thank You