Software Requirements & Specifications

Courtesy:

Roger Pressman, Ian Sommerville &

Prof Rajib Mall

Objectives of the chapter:

- To describe the principal requirements engineering activities
- To introduce the techniques for requirements elicitation and the associated work products
- To introduce the techniques for requirements analysis and the associated work products
- To describe the techniques for requirements specification, validation, verification and associated work products.

Requirement Definition:

- IEEE definition
  - A condition or a capability that must be met or processed by a system .. to satisfy a contract, standard, specification or other formally imposed document
- In general
  - the process of establishing
    - the services that the customer requires from a system and
    - the constraints under which it operates and is developed.

What is a requirement?

- It describes features and functionalities need to be implemented in the system.
- The requirements for a systems are the descriptions of the services provided by the system and its operational constraints.
- Capability or a Condition that is required from a system.

Requirements….

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
- This is inevitable (unavoidable) as requirements may serve a dual function
  - May be the basis for a bid for a contract - therefore must be open to interpretation;
  - May be the basis for the contract itself - therefore must be defined in detail;
  - Both these statements may be called requirements.

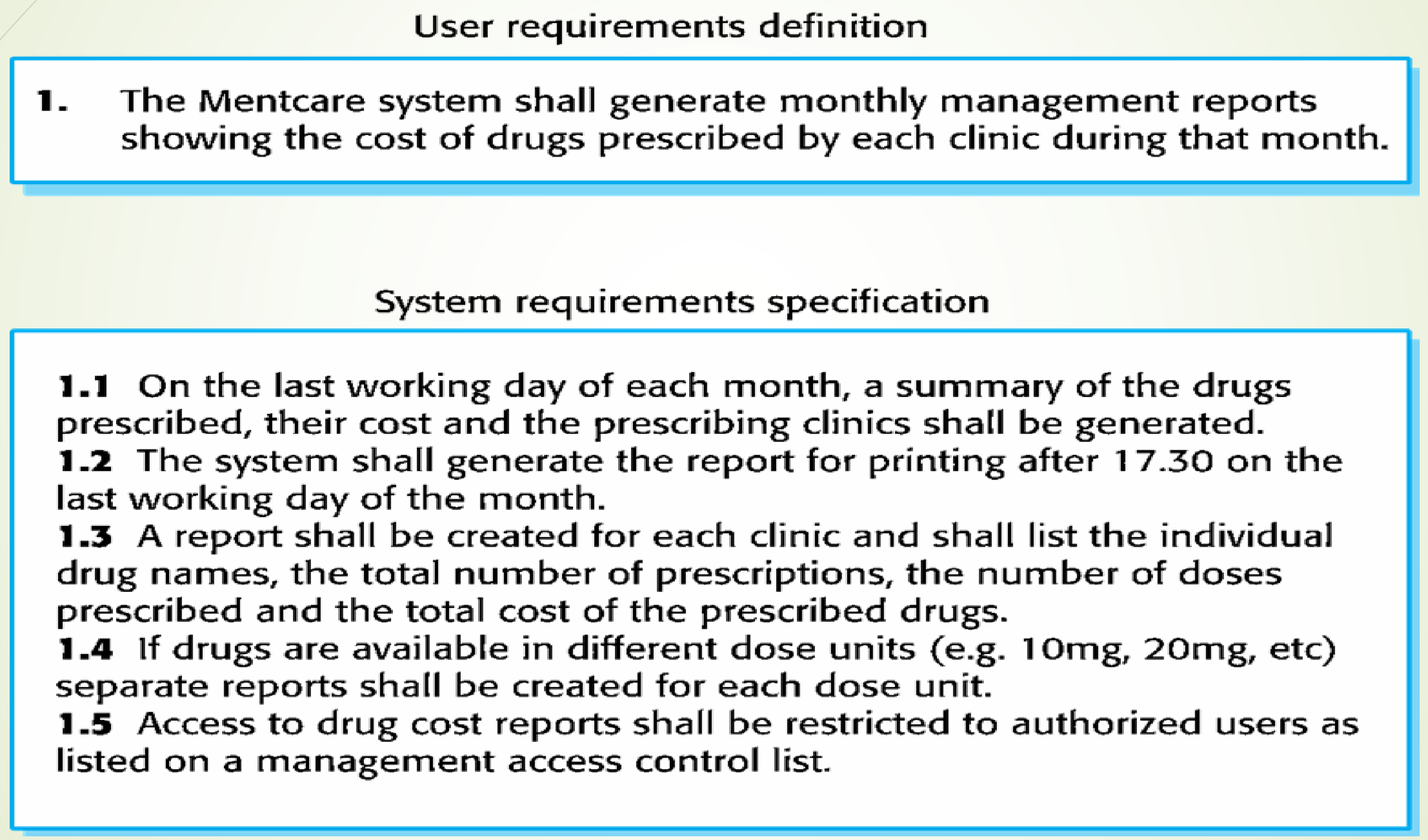
"If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organisation's needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system."

Software System Requirements…

- **User requirements:**
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.

- **System requirements:**
  - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

Software System Requirements…



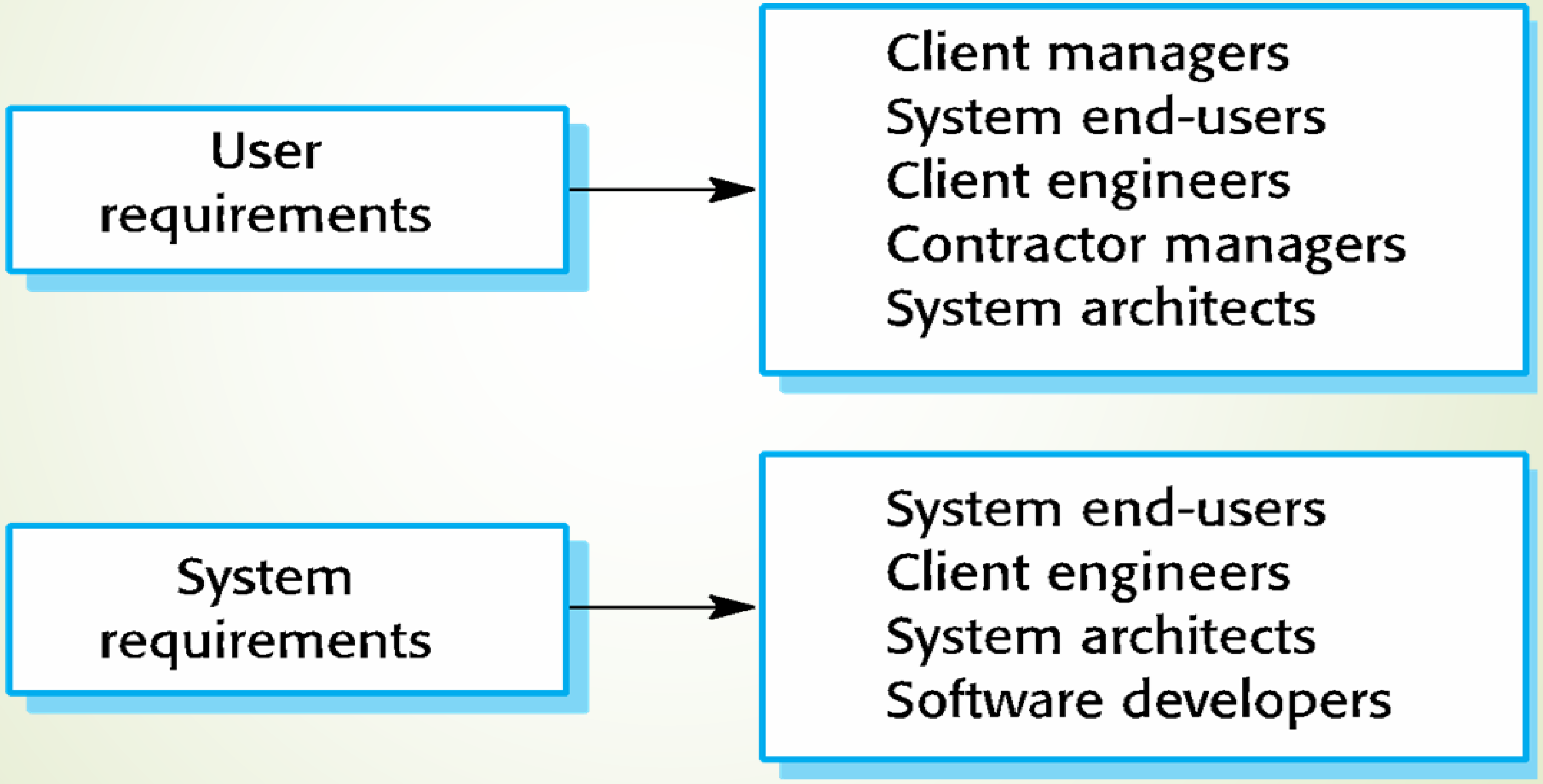
User requirements definition

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

Readers of different types of requirements specification



Types of Requirements:

- Basically software requirement is a
  - Functional or
  - Non-functional

Functional & Non-functional Requirements:

- **Functional requirements:**
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - May state what the system should not do.
- **Non-functional requirements:**
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.

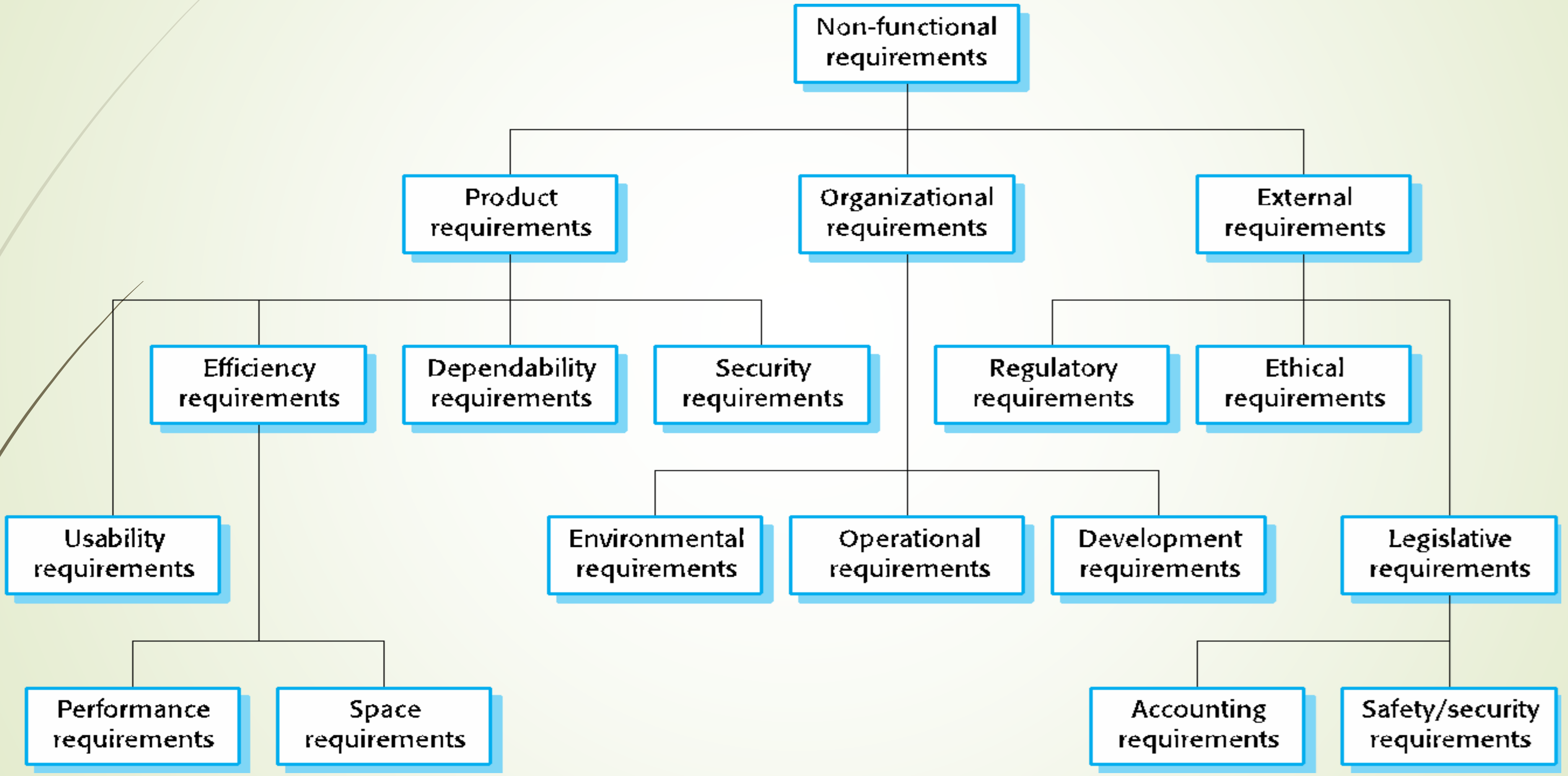
Functional Requirements:

- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do.
- Functional system requirements should describe the system services in detail.

Non-functional Requirements:

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Types of nonfunctional requirement:



Types of nonfunctional requirement:

- **Product requirements:**
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- **Organizational requirements:**
  - Requirements which are a consequence of organizational policies and procedures e.g. process standards used, implementation requirements, etc.
- **External requirements:**
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

The real problems ?

- The customer has only a vague idea of what is required
- The developer is willing to proceed with the "vague idea" on the assumption that "we will fill in the details as we go".
- The customer keep changing requirements.
- The developers is "racheted" by these changes, making errors in specifications and development.

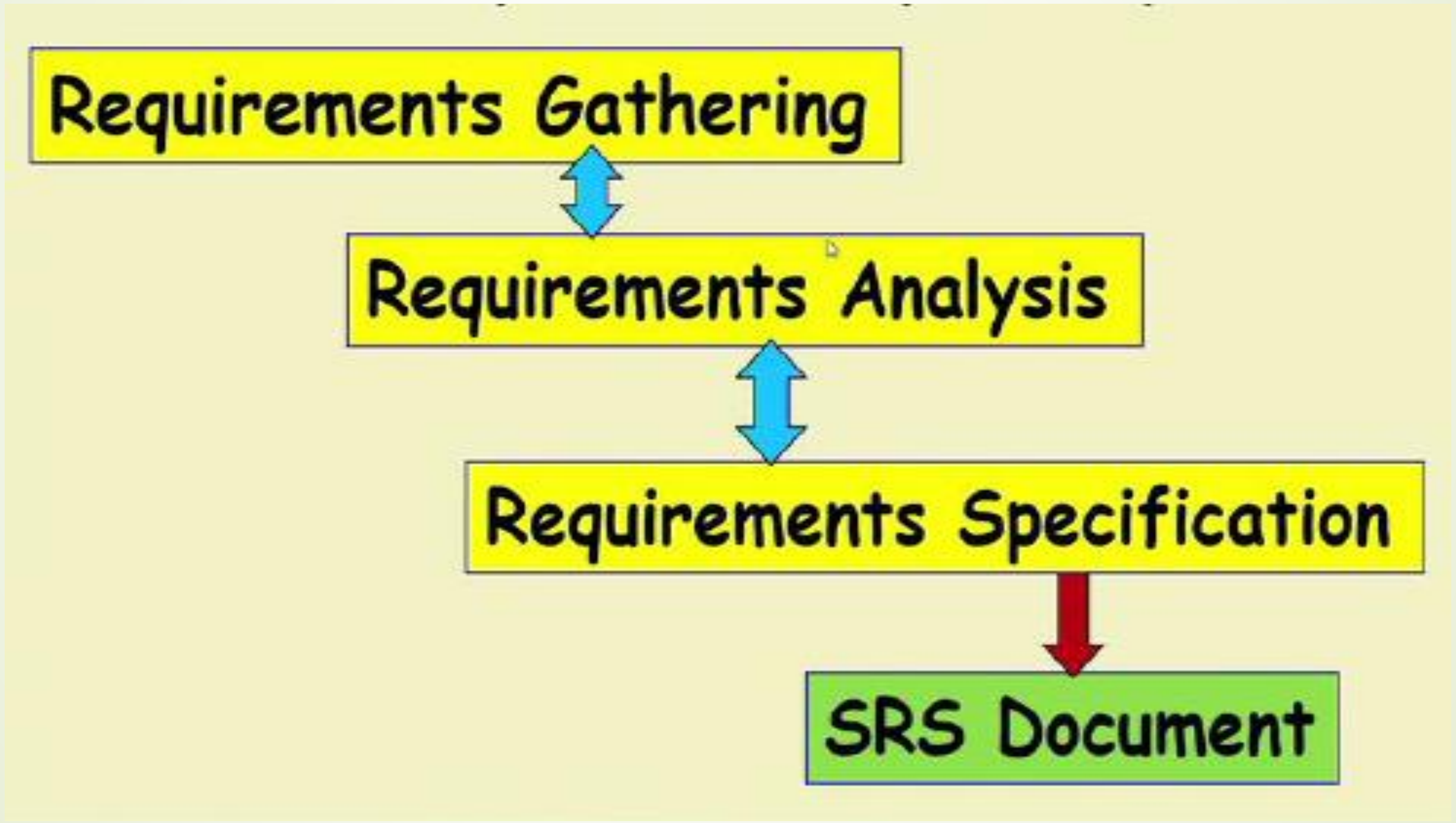
and so it goes…….

Requirements Engineering

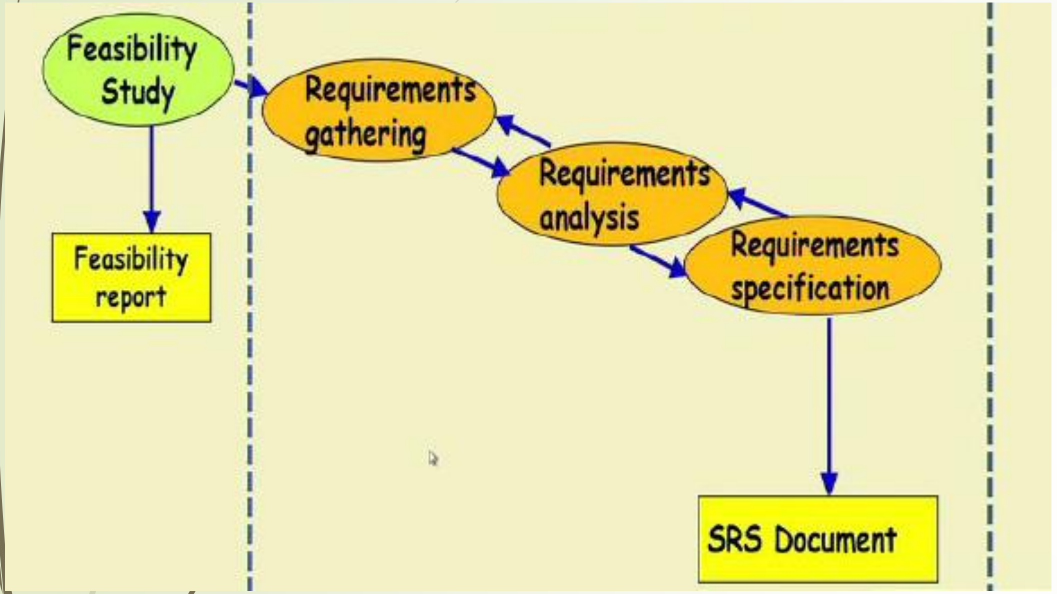
- **Requirements Engineering:** The process of finding out, analyzing, documenting and checking various services and constraints.

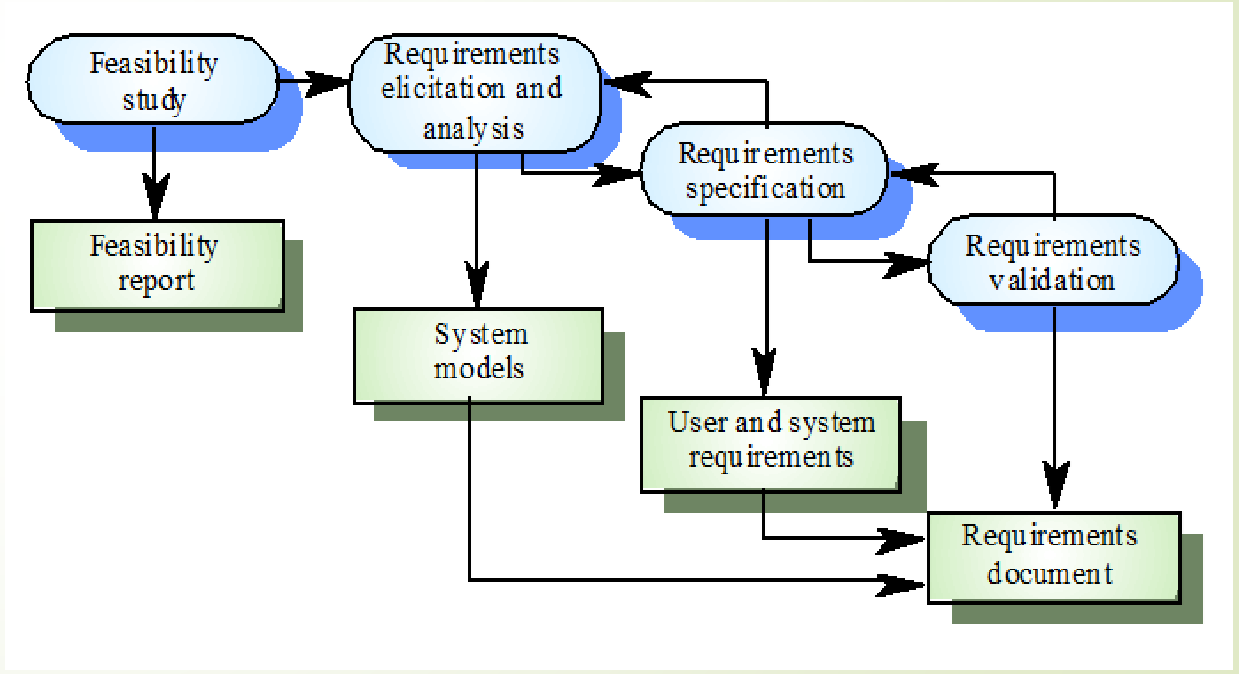
- In other words, it is the process of establishing the services that a customer requires from a system and the constraints under which it operates and is developed.

Requirement Engineering



Requirement Engineering: Elaborated





Requirement Engineering:

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

How to gather Requirements

- Observe existing systems
- Study existing procedures
- Discuss with customers and end-users
- Input and Output Analysis
- Analyze what needs to be done
- Interview
- Task Analysis
- Scenario Analysis
- Form Analysis

How to gather Requirements

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

Case Study: Automation of office work at CSE Department

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

Case Study: Automation of office work at CSE Department

- The team was first briefed by the HoD:
  - Concerning the specific activities to be automated.
- The analyst first discuss with 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 at CSE Department

- 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 tasks.
- Also collected the different types of forms that were being used.

  Form Analysis

Analysis of gathered Requirements

- Requirement analysis involves
  - Obtaining a clear, in –depth understanding of the software to be developed.
  - Remove all ambiguities and inconsistencies from the initial customer perception of the problem.
- It is quite difficult to obtain:
  - A clear, in-depth understanding of the problem:
    - Especially if there is no working model of the problem.
- Experienced systems analysts know- often as a result of painful experience---
  - "Without a clear understanding of the problem, it is impossible to develop a satisfactory system."

Analysis of gathered Requirements

- To carry out requirement analysis, the analyst need to clearly grasp the problem statement. The following questions helps to clearly understand the system:
  - What is the problem?
  - Why is it important to solve the problem?
  - What exactly are the data input to the system and what exactly are the data output by the system?
  - What are the possible procedures that need to be followed to solve the problem?
  - What are the likely complexities that might arise while solving the problem?
  - If there are external software or hardware with which the developed software has to interface, then what should be the data interchange formats with the external systems?

Analysis of gathered Requirements

- Main purpose of requirement 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 Requirements: Ambiguities

- When a requirement is ambiguous, several interpretations of that requirement are possible.
- Office Clerk: "During the final grade computation, if any student scores a sufficiently low grade in a semester, then his parents need to be informed"
- Ambiguity as there is No well defined criterion as to what can be considered as a "sufficiently low grade."
- Ex 2: Consider the term 'search'.
  - User intention: search for a student name across all departments in the institute.
  - Developer interpretation: search for a student name in an individual department. User chooses department then search.

Analysis of Requirements: Inconsistency

- Two requirements are said to be inconsistent, if one of the requirements contradicts the other.
- **Office Clerk 1:** " A student securing fail grades in three or more subjects must repeat the courses over an entire semester, and he cannot credit any other courses while repeating the courses"
- **Office Clerk 2:** "There is no provision for any student to repeat a semester; the student should clear the subject by taking it as an extra subject in any later semester."

Analysis of Requirements: Incompleteness

- An incomplete set of requirements is one in which some requirements have been overlooked.
- Incompleteness is caused by the inability of the customer to visualize the system that is to be developed.
- Office Clerk: "If a student secure a grade point average of less than 6, then the parents of the student must be intimated about the regrettable performance through a (postal) letter as well as through e-mail"

The feature that would allow entering the email ids and postal address of the student parents is missing.

Case Study: Automation of office work at CSE Department

- 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 Software Requirement Specification (SRS) document.

Software Requirements Specification (SRS)

- Once the analyst gathered all the information regarding software to be developed and has removed all incompleteness, inconsistency and ambiguity from the specification, he starts to systematically organize requirements in the form of an SRS document.
- *"A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development."*
- *SRS main aim:*
  - *Systematically organize the requirements arrived during requirements analysis.*
  - *Document requirements properly*

Software Requirements Specification (SRS)

- The SRS
  - fully describes what the software will do and how it will be expected to perform.
  - minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost.
  - defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations.
  - defines and evaluates the parameters such as operating speed, response time, availability, portability, maintainability, security and speed of recovery from adverse events are evaluated.
  - SRS document is expected to cater to the needs of a wide variety of audience.

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

Characteristics of good SRS

- It should be Concise
- Implementation independent: It should specify what the system must do
  - And not say how to do
- Easy to change
- Consistent
- Complete
- Traceable
- Verifiable
- Identification of response to undesired events

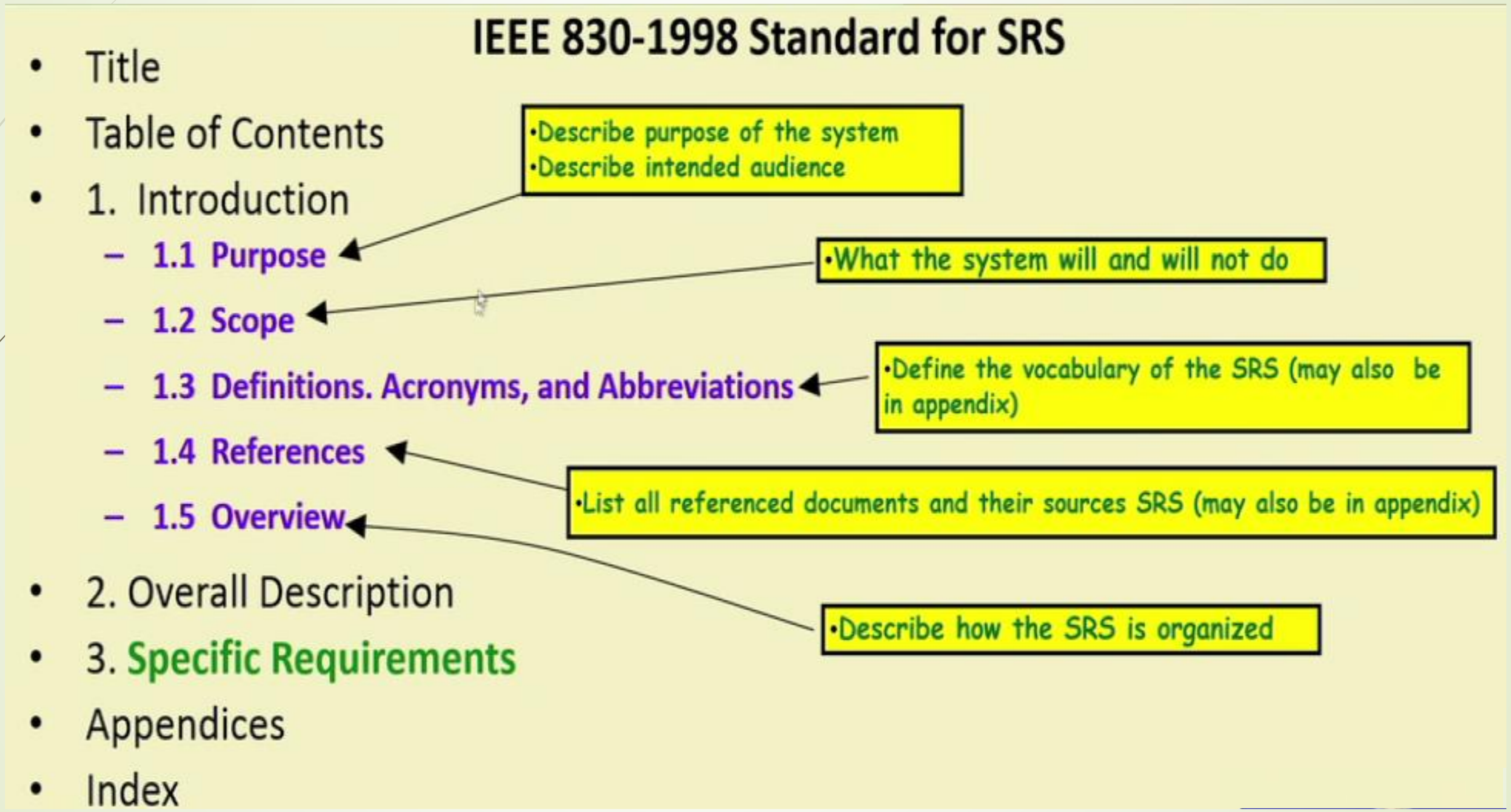
Attributes of good SRS

- readable and understandable by customers, users, and designers.
- structured to be easy to change.
- be able to serve as a reference for system maintainers.
- consistent, complete, unambiguous, realistic & testable
- Specifies
  - only external system behavior (black box)
  - both goals and constraints.
  - acceptable responses to undesired events.
  - what should not be done as well as what should be done.
  - incremental subsets if desired or minimum and maximum functionality
  - changes anticipated in the future (for both environment and software)

Contents of SRS:

- Different templates for requirements specifications used by companies: Often variations of IEEE 830
- Four important parts:
  - **Functional requirements**
    - Specifies all the functionality that the system should support.
  - **External Interfaces**
    - User interface, Hardware interface, software interface, communication interface with other systems, File export format.
  - **Non-functional requirements**
    - Maintainability, Portability, Usability, Security, Safety, reliability, performance, etc.
  - **Constraints**
    - Hardware to be used, Operating System or DBMS to be used, Capabilities of I/O devices.

Contents of SRS:



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

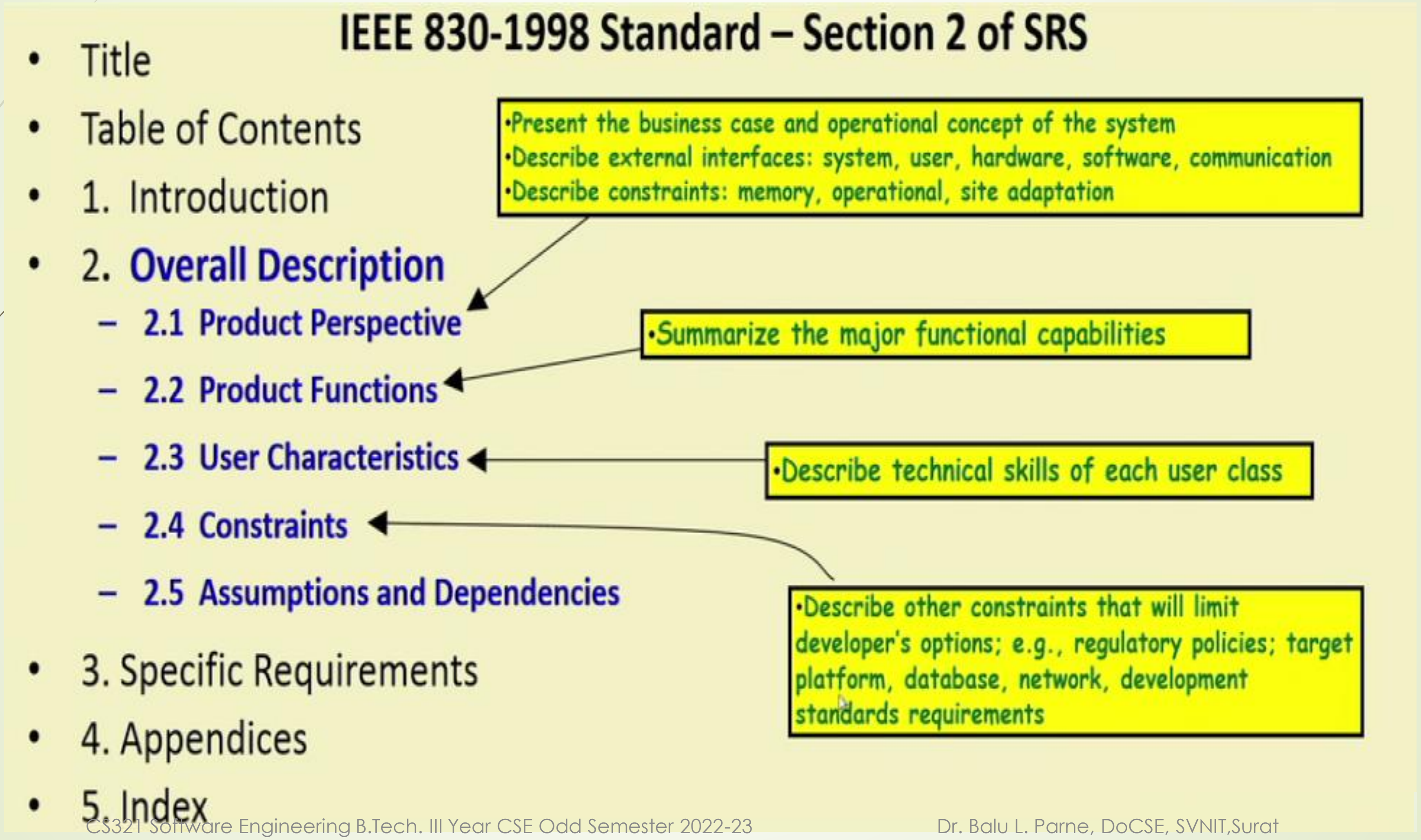
•What the system will and will not do

•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

Contents of SRS:



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

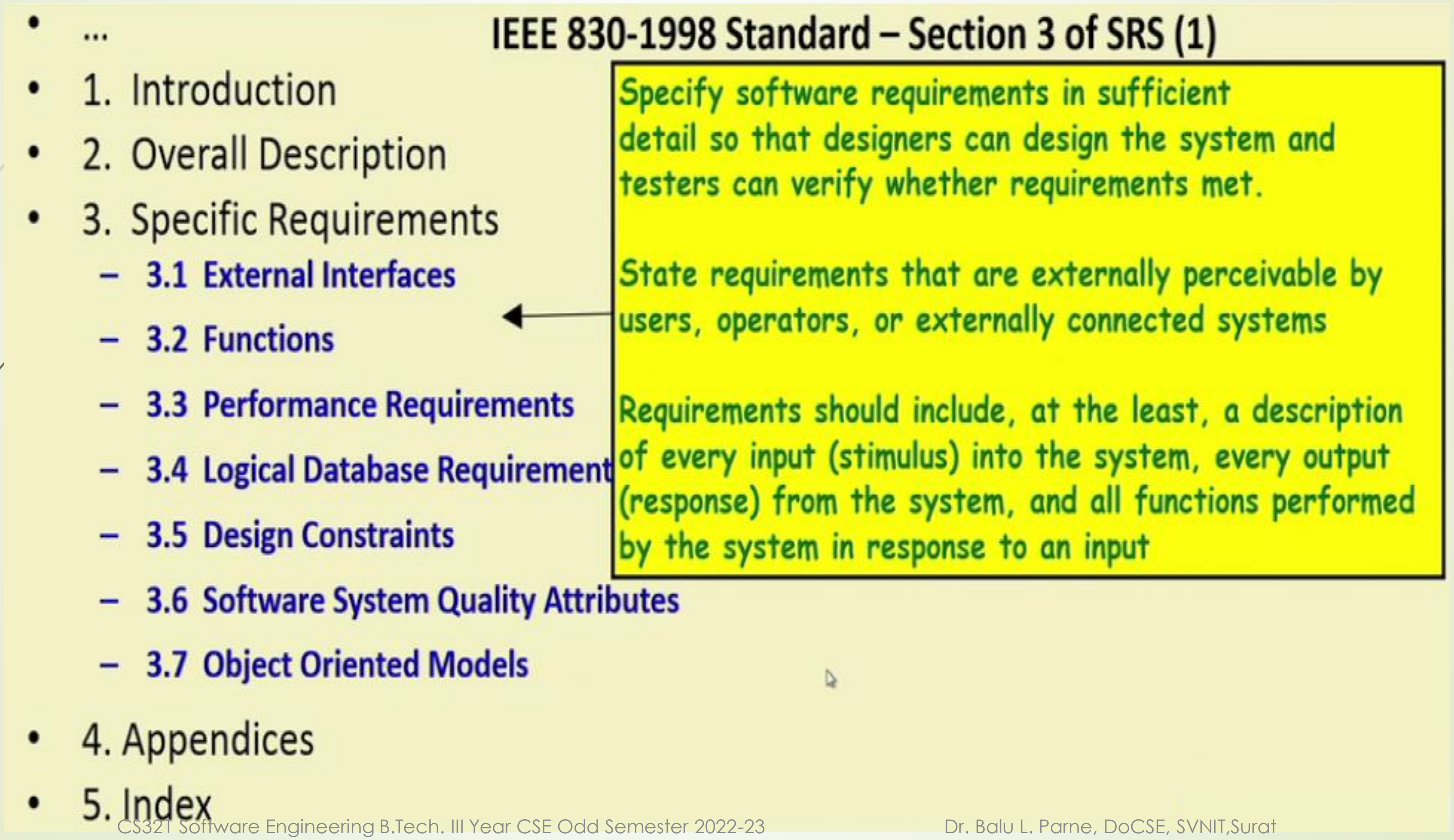
•Present the business case and operational concept of the system
•Describe external interfaces: system, user, hardware, software, communication
•Describe constraints: memory, operational, site adaptation

•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

Contents of SRS:



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

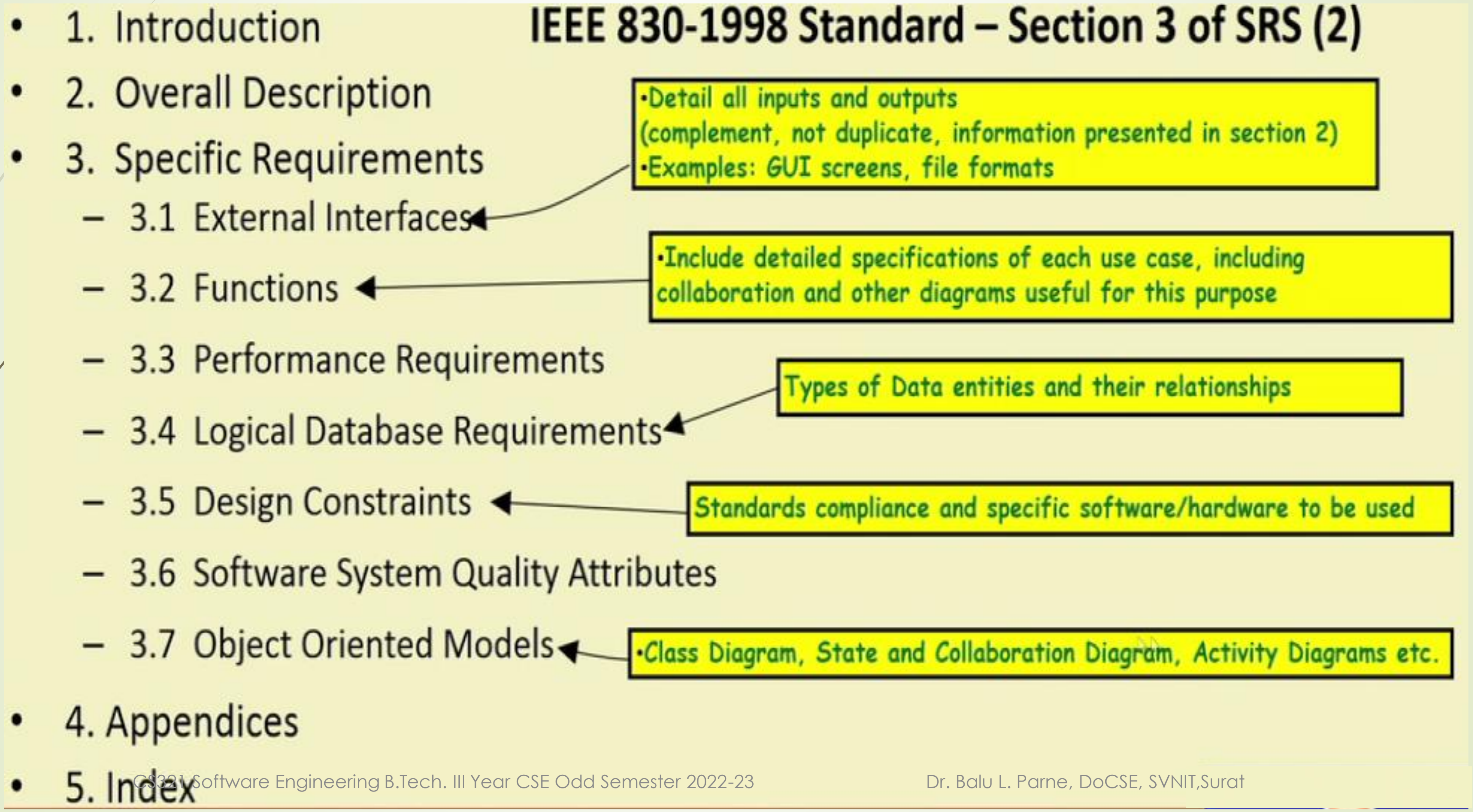
- ...
- 1. Introduction
- 2. Overall Description
- 3. Specific Requirements
  - 3.1 External Interfaces
  - 3.2 Functions
  - 3.3 Performance Requirements
  - 3.4 Logical Database Requirement
  - 3.5 Design Constraints
  - 3.6 Software System Quality Attributes
  - 3.7 Object Oriented Models
- 4. Appendices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

Contents of SRS:



IEEE 830-1998 Standard – Section 3 of SRS (2)

- 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

•Detail all inputs and outputs
(complement, not duplicate, information presented in section 2)
•Examples: GUI screens, file formats

•Include detailed specifications of each use case, including collaboration and other diagrams useful for this purpose

Types of Data entities and their relationships

Standards compliance and specific software/hardware to be used

•Class Diagram, State and Collaboration Diagram, Activity Diagrams etc.

Contents of SRS:

- **Introduction section**
  - **Purpose:** describes where the software would be deployed and how the software would be used.
  - **Project Scope:** describes the overall context within which the software is being developed.
  - **Environmental Characteristics:** briefly outline the environment (hardware and other software) with which the software will interact.

Contents of SRS:

- **Overall description of organization of SRS document:**
  - **Product perspective:** briefly state as to whether the software is intended to be a replacement for certain existing system, or it is a new software.
  - **Product Features:** Summarize the major ways in which the software would be used.
  - **User Classes:** Various user classes that are expected to use this software are identified and described.
  - **Operating environment:** discuss the hardware platform on which the software would run, the operating system and other application software with which the developed software would interact.

Contents of SRS:

- **Functional Requirements**
  - Classify the functionalities either based on the specific functionalities invoked by different users, or functionalities that are available in different modes.
    - User class 1
      - (a) Functional requirement 1.1
      - (b) Functional requirement 1.2
    - User class 2
      - (a) Functional requirement 2.1
      - (b) Functional requirement 2.2

Contents of SRS:

- **External Interface Requirements:**
  - **User interface:** Include sample screen images, any GUI standards or style guides that can be followed, screen layout constraints, etc.
  - **Hardware interface:** Description of the supported device types, the nature of data and control interactions between the software and the hardware, and the communication protocols to be used.
  - **Software interface:** Describes the connection between this software and other specific software components, including databases, operating systems, tools, libraries, etc.
  - **Communications interface:** Describes the requirements associated with any type of communications required by the software, such as email, web access, network server communications protocols, etc.

Contents of SRS:

- **Other non-functional requirements:**
  - **Performance requirements:** Aspects such as number of transaction to be completed per second should be specified here.
  - **Safety requirements:** Aspects such as recovery after failure, handling software and hardware failure.
  - **Security requirements:** specify security and privacy requirements on data used or created by the software.
- **Glossary**
  - Definitions of technical terms used in the document.
- **Indexes**
  - Various types of indexes may be provided.

Contents of SRS:

- https://www.freestudentprojects.com/studentprojectreport/project-srs/library-management-system-project-srs-document/
- http://mca.ignougroup.com/2017/01/srs-for-library-management-system.html

Functional Requirements: Academic Administration Software

- **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: Academic Administration Software

- **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: Academic Administration Software

- **3.3 Non-Functional Requirements**
  - **N-001:** AAS Shall respond to query in less than 5 seconds.
  - **N-002:** AAS shall operate with zero downtime.
  - **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.

Functional Requirements contd…

- It is desirable to consider every system as:
  - Performing a set of functions {fi}
- Each function fi considered as:
  - Transforming a set of input data to corresponding output data
- Example 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 Requirements contd…

- Functional requirements describe:
  - A set of high-level requirements
  - Each high-level requirement:
    - Takes in some data from the user
    - Outputs some data to the user
    - Might consist of a set of identifiable sub-functions
  - For each high-level requirement:
    - A function is described in terms of:
      - Input data set
      - Output data set
      - Processing required to obtain the output data set from the input data set

Functional Requirements contd…

- Even for the same high level function
  - There can be different interaction sequences (or scenarios)
  - Due to users selecting different options or entering different data items.
- **Use Cases:**
  - A use case is term in UML:
    - Represents a high level functional requirement.
  - Use case representation is more well-defined and has agreed documentation:
    - Compared to high level functional requirements and its documentation
    - Therefore many organizations document the functional requirements in term of use case

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

Example of Bad SRS Documents:

- **Over specification:**
  - 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 to mean different things.
- **Ambiguity:**
  - Literary expressions
  - Unquantifiable aspects, e.g. "good user interface"

Example of Bad SRS Documents:

- **Wishful Thinking:**
  - Description of aspects
  - For which realistic solutions will be hard to find.
- **Forward References:**
  - References to aspects of problem
  - Defined only later on in the text.

Suggestion for Writing Good Quality SRS Documents:

- Keep sentences and paragraphs short.
- Use active voice.
- Use proper grammar, spelling, and punctuation.
- Use terms consistently and define them in glossary.
- Split a requirement into multiple sub-requirements:
  - Because each will require separate test cases and each should be separately traceable.
  - If several requirements are combine together in a paragraph, it is easy to overlook some during development or testing.
- To see if a requirement statement is sufficiently well defined,
  - Read it from the customers and developers perspective

Use of SRS:

- Establishes the basis for agreement between the customers and the suppliers/developers.
- 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 or future extensions.

Why SRS:

- SRS establishes basis of agreement between the user and the supplier.
  - Users needs have to be satisfied, but user may not understand software
  - Developers will develop the system, but may not know about problem domain
  - SRS is the medium to bridge the communication gap and specify user needs in a manner both can understand

Why SRS:

- Helps user understand his needs.
  - users do not always know their needs
  - must analyze and understand the potential
  - the goal is not just to automate a manual system, but also to add value through it.
  - The requirements process helps clarify needs
- SRS provides a reference for validation of the final product
  - Clear understanding about what is expected.
  - Validation - "SW satisfies the SRS"

Why SRS:

- Good SRS reduces the development cost
  - SRS errors are expensive to fix later
  - Req. changes can cost a lot (up to 40%)
  - Good SRS can minimize changes and errors
  - Substantial savings; extra effort spent during req. saves multiple times that effort

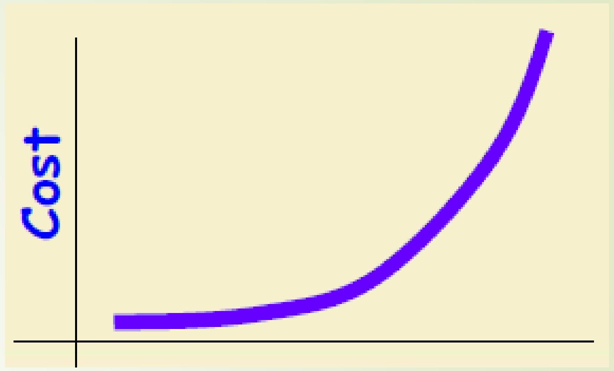
Why SRS:

- An Example
- Cost of fixing errors….
  - in req., design , coding , acceptance testing and operation are 2 , 5 , 15 , 50 , 150 person-hours
- After req. phase,
  - 65% requirements errors detected in design , 2% in coding, 30% in Acceptance testing, 3% during operation
- If 50 requirement errors are NOT removed in the req. phase, what is the total cost in –personhours?
  - 32.5*5+1*15+15*50+1.5*150=1152 hours
- What if one spends more person-hours on requirements ?.....

Why SRS:

- An Example……
  - If 100 person-hours invested additionally in requirement to catch these 50 defects , what is the advantage?
  - What is the net reduction in costs? …..
  - the development cost could be reduced by 1152 person-hours.
  - Net reduction in cost is 1052 person-hours

Need for SRS

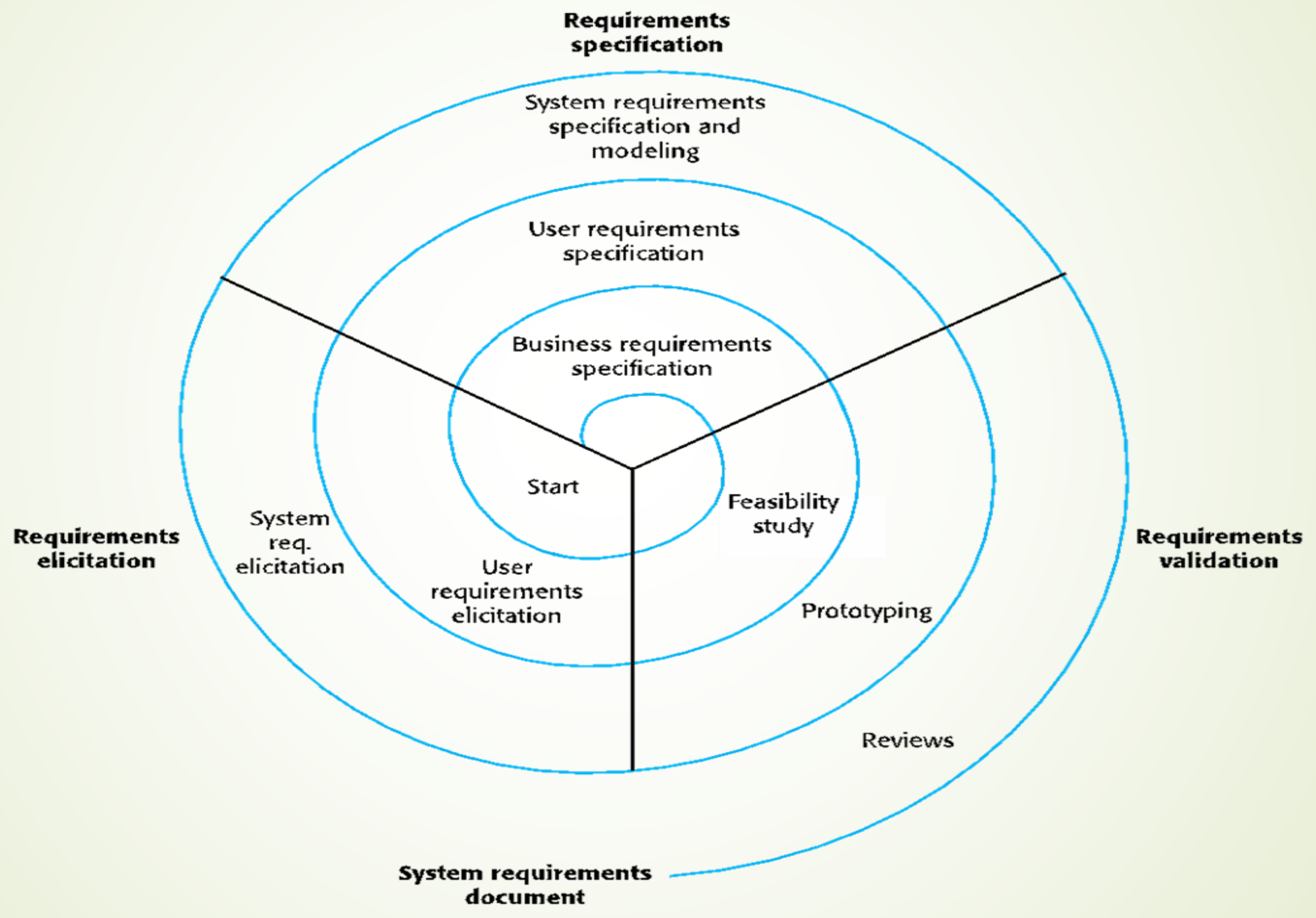


- Good SRS reduces development cost:
  - Requirement errors are expensive to fix later
  - Requirement changes cost a lot (typically 40% of requirements change later)
  - Good SRS can minimize changes and errors
  - Substantial savings ---effort spent during req. saves multiple times that effort
- Cost of fixing errors in req. , design , coding , testing and operation increases exponentially

Requirements Engineering Processes

- The processes used for RE vary widely depending on the application domain, the people involved and the organization developing the requirements.
- However, there are a number of generic activities common to all processes
  - Requirements elicitation;
  - Requirements analysis;
  - Requirements validation;
  - Requirements management.
- In practice, RE is an iterative activity in which these processes are interleaved.

A spiral view of the RE process


Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Requirements elicitation
System req. elicitation
User requirements elicitation
Requirements validation
Prototyping
Reviews
System requirements document

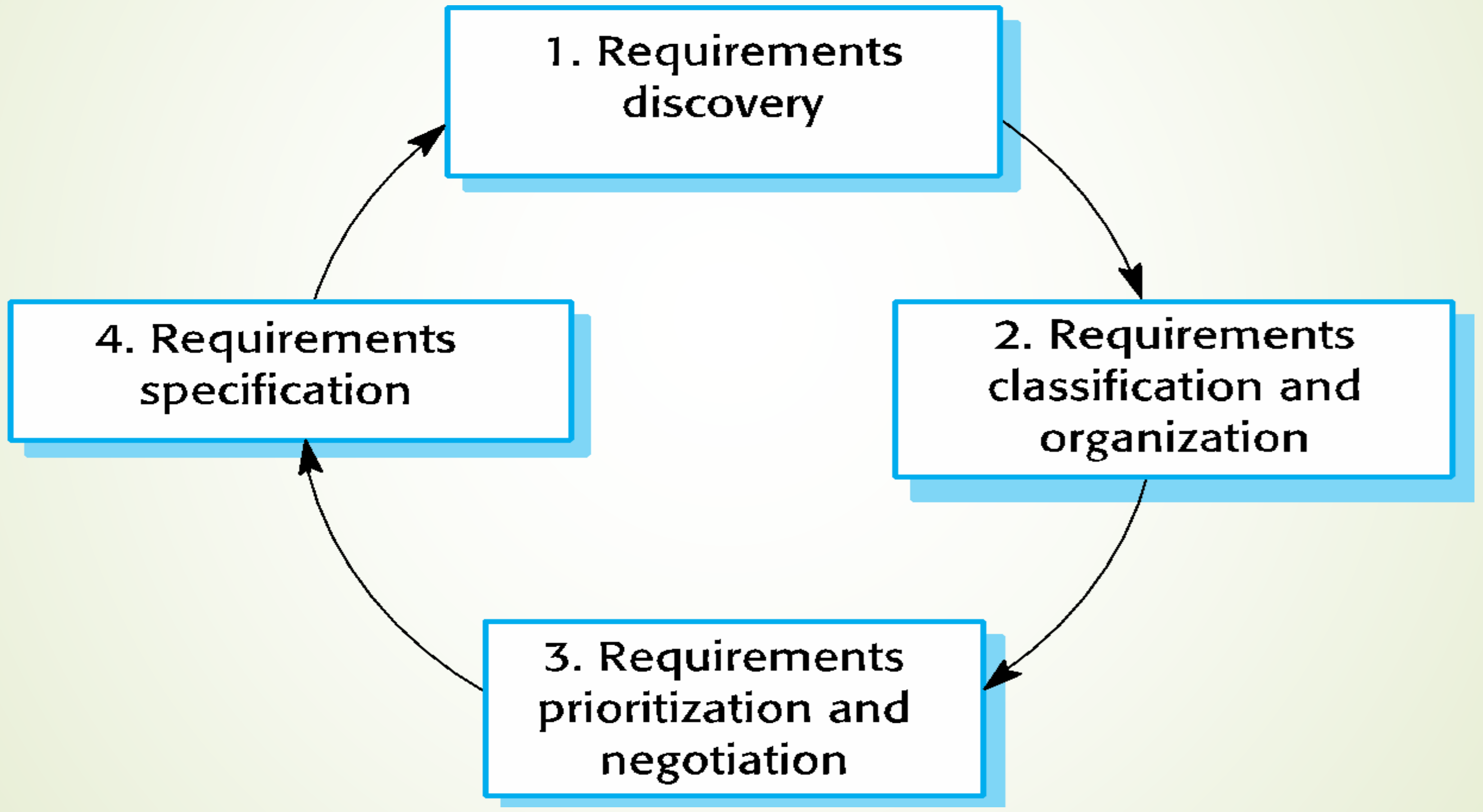
Requirements elicitation

- Sometimes called requirements elicitation or requirements discovery.
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved in maintenance, domain experts, etc. These are called stakeholders.

Requirements elicitation

- Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
- Stages include:
  - Requirements discovery,
  - Requirements classification and organization,
  - Requirements prioritization and negotiation,
  - Requirements specification.

The requirements elicitation and analysis process:


1. Requirements discovery
2. Requirements classification and organization
3. Requirements prioritization and negotiation
4. Requirements specification

Process activities:

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

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.
- Organizational 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.

Requirements Specification

- The process of writing down the user and system requirements in a requirements document.
- User requirements have to be understandable by end-users and customers who do not have a technical background.
- System requirements are more detailed requirements and may include more technical information.
- The requirements may be part of a contract for the system development
  - It is therefore important that these are as complete as possible.

Ways of writing a system requirements specification:

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract

Requirements validation:

- Concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

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?

Requirements validation techniques

- Requirements reviews
  - Systematic manual analysis of the requirements.
- Prototyping
  - Using an executable model of the system to check requirements.
- Test-case generation
  - Developing tests for requirements to check testability.

Requirements reviews:

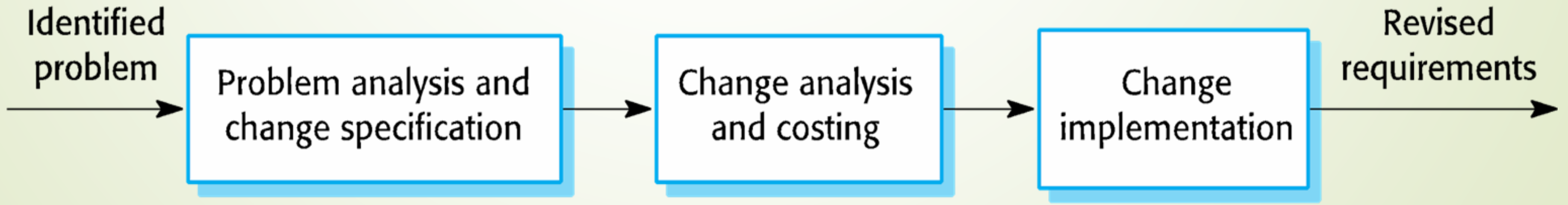
- Regular reviews should be held while the requirements definition is being formulated.
- Both client and contractor staff should be involved in reviews.
- Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage.
- Review checks:
  - Verifiability
    - Is the requirement realistically testable?
  - Comprehensibility
    - Is the requirement properly understood?
  - Traceability
    - Is the origin of the requirement clearly stated?
  - Adaptability
    - Can the requirement be changed without a large impact on other requirements?

Requirements change:

- **Reasons for Requirement Change:**
  - The business and technical environment of the system always changes after installation.
  - The people who pay for a system and the users of that system are rarely the same people.
  - Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.

Requirements management:

- Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- New requirements emerge as a system is being developed and after it has gone into use.
- You need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes. You need to establish a formal process for making change proposals and linking these to system requirements.


Identified problem
Problem analysis and change specification
Change analysis and costing
Change implementation
Revised requirements

Key Points: Summary

- **Requirements** for a software system set out what the system should do and define constraints on its operation and implementation.
- **Functional requirements** are statements of the services that the system must provide or are descriptions of how some computations must be carried out.
- **Non-functional requirements** often constrain the system being developed and the development process being used.
- They often relate to the emergent properties of the system and therefore apply to the system as a whole.

Key Points: Summary

- The **requirements engineering process** is an iterative process that includes requirements elicitation, specification and validation.
- **Requirements elicitation** is an iterative process that can be represented as a spiral of activities – requirements discovery, requirements classification and organization, requirements negotiation and requirements documentation.
- You can use a range of techniques for requirements elicitation including interviews and ethnography. User stories and scenarios may be used to facilitate discussions.

Key Points: Summary

- **Requirements specification** is the process of formally documenting the user and system requirements and creating a software requirements document.
- **Requirements validation** is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.
- Business, organizational and technical changes inevitably lead to changes to the requirements for a software system. **Requirements management** is the process of managing and controlling these changes.

Thank You.