Shri Vaishnav Vidyapeeth Vishwavidyalaya

Department of Computer Science & Engineering

Subject:- SEPM

Subject Code:- BTCS504

Class:- 3Year CS - C, D, E & H

Ms. Mandakini Ingle
Assistant Professor
CSE

SYLLABUS

Unit II: Requirement Elicitation, Analysis, and Specification

- **Functional and Non-functional Requirements:** Requirement Sources and Elicitation Techniques,
- Analysis Modeling for Function-oriented and Object-oriented Software Development,
- Use case Modeling,
- System and Software Requirement Specifications,
- Requirement Validation, Traceability

Software Engineering- Requirement Elicitation and Specification

- **Requirements:-**
- Features of system or system function used to fulfill system purpose
- Focus on customer's needs and problem, not on solutions
- Condition or capability

Types of 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

Requirements Engineering

- The process to gather the software requirements from client, analyze and document them is known as requirement engineering.
- It is a Systematic use of principle,techniques & tools.
- The goal of requirement engineering is to develop and maintain sophisticated and descriptive ‘System Requirements Specification’ document.
- S/W developer & Client both can take active role in requirement engineering

Requirements Engineering,,,.........

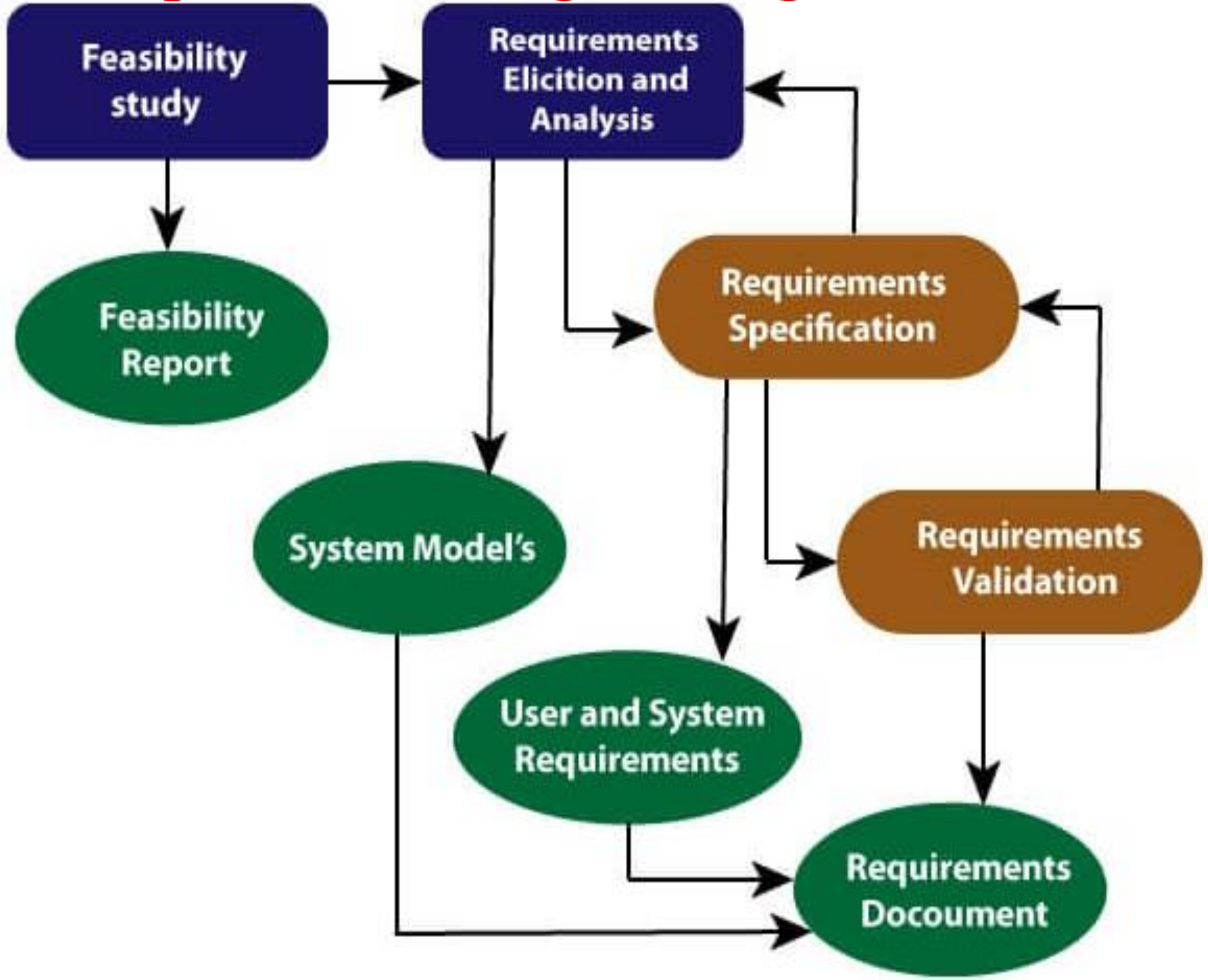
- Requirement Engineering is the process of defining, documenting and maintaining the requirements.
- It is a process of gathering and defining service provided by the system.
- Requirements Engineering Process consists of the following main activities:

Requirements Engineering,,,..........

Requirement Engineering Process

1. **Feasibility Study**
2. **Requirement Elicitation and Analysis**
3. **Software Requirement Specification**
4. **Software Requirement Validation**
5. **Software Requirement Management**

Requirements Engineering Process



Requirements Engineering,,,……….

1. Feasibility Study:

The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.

Types of Feasibility:

1. **Technical Feasibility** - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.
2. **Operational Feasibility** - Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.
3. **Economic Feasibility** - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

Requirements Engineering,,,.........

2. Requirement Elicitation and Analysis:

This is also known as the gathering of requirements. Here, requirements are identified with the help of customers and existing systems processes, if available.

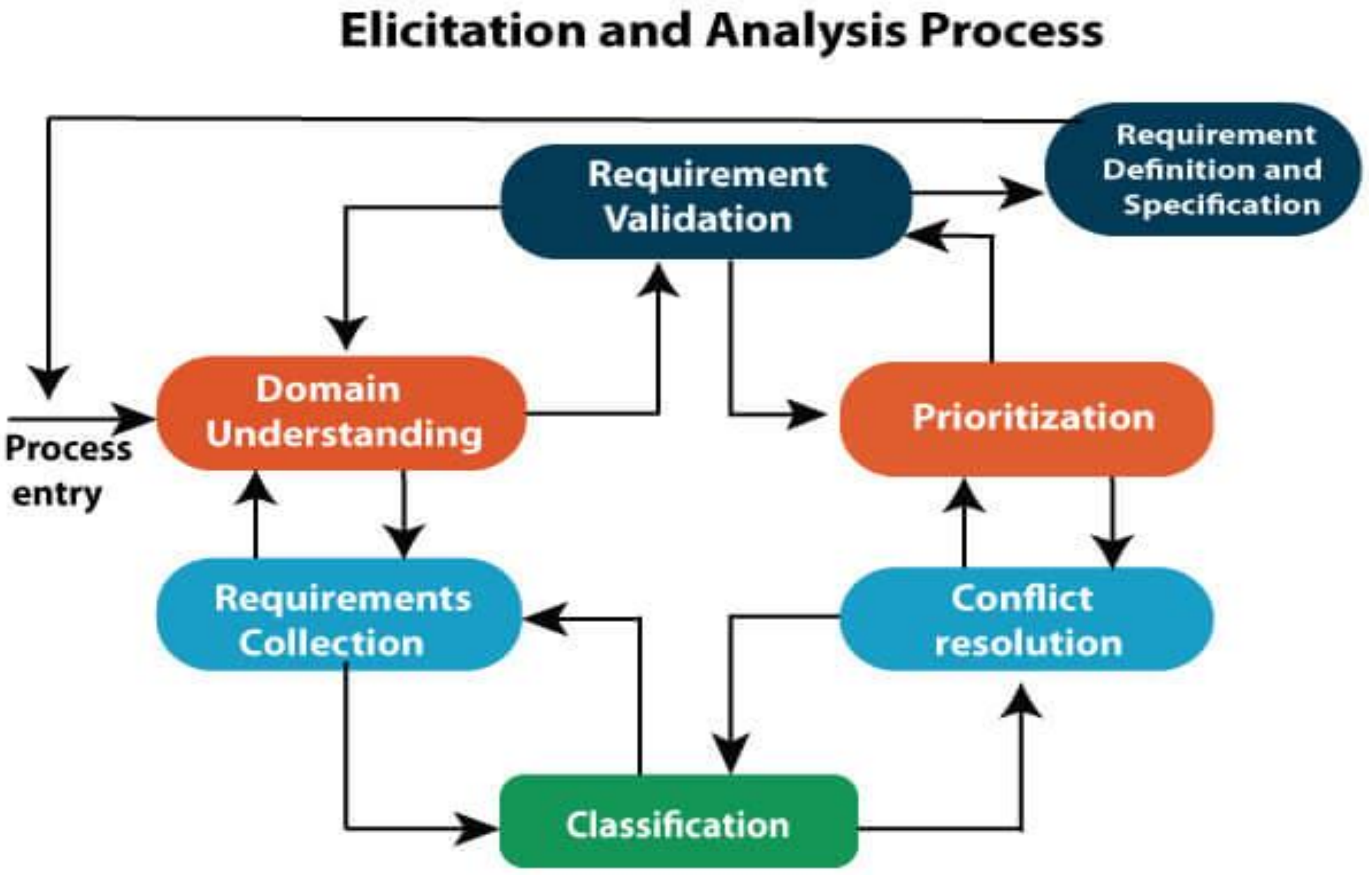
Analysis of requirements starts with requirement elicitation. The requirements are analyzed to identify inconsistencies, defects, omission, etc. We describe requirements in terms of relationships and also resolve conflicts if any.

Problems of Elicitation and Analysis

1. Getting all, and only, the right people involved.
2. Stakeholders often don't know what they want
3. Stakeholders express requirements in their terms.
4. Stakeholders may have conflicting requirements.
5. Requirement change during the analysis process.
6. Organizational and political factors may influence system requirements.

Requirements Engineering,,,.........

Department of Computer Science and Engineering



Requirements Engineering,,,.........

3. Software Requirement Specification:

Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language. It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.

The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.

Requirements Engineering,,,..........

Data Flow Diagrams: Data Flow Diagrams (DFDs) are used widely for modeling the requirements. DFD shows the flow of data through a system. The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding. The DFD is also known as a data flow graph or bubble chart.

Data Dictionaries: Data Dictionaries are simply repositories to store information about all data items defined in DFDs. At the requirements stage, the data dictionary should at least define customer data items, to ensure that the customer and developers use the same definition and terminologies.

Requirements Engineering,,,..........

Entity-Relationship Diagrams: Another tool for requirement specification is the entity-relationship diagram, often called an "E-R diagram." It is a detailed logical representation of the data for the organization and uses three main constructs i.e. data entities, relationships, and their associated attributes.

Requirements Engineering,,,..........

4. Software Requirement Validation:

After requirement specifications developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs. Requirements can be the check against the following conditions -

- If they can practically implement
- If they are correct and as per the functionality and specially of software
- If there are any ambiguities
- If they are full
- If they can describe

Requirements Engineering,,,.........

5. Requirements Validation Techniques

1. Requirements reviews/inspections: systematic manual analysis of the requirements.
2. Prototyping: Using an executable model of the system to check requirements.
3. Test-case generation: Developing tests for requirements to check testability.
4. Automated consistency analysis: checking for the consistency of structured requirements descriptions.

Software Requirements:

Functional Requirements: Functional requirements define a function that a system or system element must be qualified to perform and must be documented in different forms. The functional requirements are describing the behavior of the system as it correlates to the system's functionality.

Non-functional Requirements: This can be the necessities that specify the criteria that can be used to decide the operation instead of specific behaviors of the system.

Non-functional requirements are divided into two main categories:

Execution qualities like security and usability, which are observable at run time.

Evolution qualities like testability, maintainability, extensibility, and scalability that embodied in the static structure of the software system.

Requirements Engineering,,,..........

6. Software Requirement Management:

Requirement management is the process of managing changing requirements during the requirements engineering process and system development.

New requirements emerge during the process as business needs a change, and a better understanding of the system is developed.

The priority of requirements from different viewpoints changes during development process.

The business and technical environment of the system changes during the development.

Requirements elicitation Techniues:

- A technique to find out the Requirement for intended s/w communica?ting by clien,end user,system user, & other who have a stake in sopftware system development activity tounderstand the requirement.
- Also called Requirement Discovery.
- Discover Each & every reuirements.
- Critcal aspect of s/w
- This is Performed after receiving the problem statement.

Requirements elicitation Techniues.....

- Knowledge of the overall area where the systems is applied.
- The details of the precise customer problem where the system are going to be applied must be understood.
- Interaction of system with external requirements.
- Detailed investigation of user needs.
- Define the constraints for system development.

Requirements elicitation Techniques:

1. Interviews
2. Brainstorming Sessions
3. Facilitated Application Specification Technique (FAST)
4. Quality Function Deployment (QFD)
5. Use Case Approach
6. Questionaries
7. Domain Analysis
8. Observation
9. Prototyping

The success of an elicitation technique used depends on the maturity of the analyst, developers, users, and the customer involved.

Requirements elicitation Techniues:

1. Interviews:-

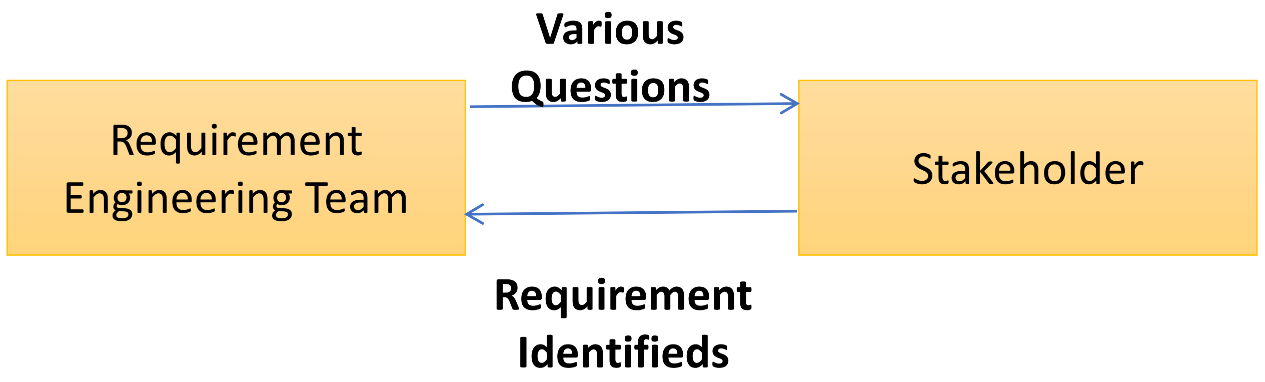
- ❖ Objective of conducting an interview is to understand the customer's expectations from the software.
- ❖ It is impossible to interview every stakeholder hence representatives from groups are selected based on their expertise and credibility.
- ❖ Getting an overall understanding "What" stakeholders do,how they interact with system & difficulties they faced with curent system.
- ❖ It is an effective & traditional mean of information.
- ❖ Interviews maybe formal & Informal.

Requirements elicitation Techniues:

Interviews maybe be formal & Informal.

Interviews maybe be open-ended or structured.

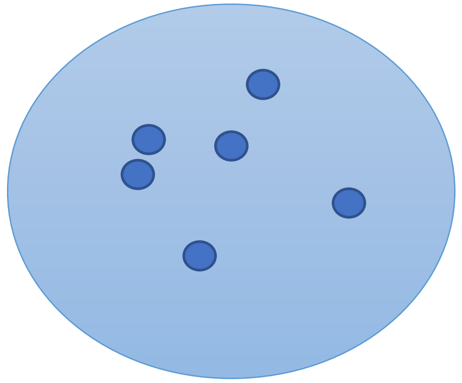
1. In open-ended interviews there is no pre-set agenda. Context free questions may be asked to understand the problem.
2. In structured/CLOSED interview, agenda of fairly open questions is prepared. Sometimes a proper questionnaire is designed for the interview.
3. Oral Interview
4. Written Interview
5. ONe to One
6. Group interview



Charactristics of effective interview

1. Interviews Conduct in open environment.
2. Should be condected open minded.
3. Interviews Should be starts with discussion by asking questions

Requirements elicitation Techniues:



2. Brainstorming Sessions-

It is a group technique

Exchange of specific ideas

It is intended to generate lots of new ideas hence providing a platform to share views

A highly trained facilitator is required to handle group bias and group conflicts.

Every idea is documented so that everyone can see it.

Finally, a document is prepared which consists of the list of requirements and their priority if possible.

Requirements elicitation Techniues:

2. Brainstorming Sessions………

Informal debate helding among many developers and customers

It is a group discussion

Creative technique & new ideas.

Platform to express and share views expectations & difficulties.

Conflicts are removed

Additional ideas

Relevant & irrelevant ideas also included

Requirements elicitation Techniues:

3. Questionaries-

1. A document with the predefined set of objective questions & respective options is handed over to all stakeholder.
2. The answer are collected and compiled
3. Large no. of people
4. less time and less cost.it should be clearly analyzed. APPLY systematic & Quantitative methods

Result from user depend on 2 factors

1. Effective and design of questionaries

2. Honesty of respond

Requirements elicitation Techniues:

4. Facilitated Application Specification Technique: [FAST]

It’s objective is to bridge the expectation gap – difference between what the developers think they are supposed to build and what customers think they are going to get.

A team oriented approach is developed for requirements gathering.

Each attendee is asked to make a list of objects that are-

1. Part of the environment that surrounds the system
2. Produced by the system
3. Used by the system

Requirements elicitation Techniues:

4. Facilitated Application Specification Technique:

Each participant prepares his/her list, different lists are then combined, redundant entries are eliminated, team is divided into smaller sub-teams to develop mini-specifications and finally a draft of specifications is written down using all the inputs from the meeting.

Requirements elicitation Techniues:

These basic guidelines.

•A meeting is conducted at a neutral site and attended by both software engineering analysts and customers.

•Rules for preparation and participation are established.

•An agenda is suggested that is formal enough to cover all important point but informal enough to encourage the free flow of ideas.

•A facilitator (can be a customer a developer or an outsider) is assigned to control the meeting.

•A definition mechanism (can be work sheets flip charts or wall stickers or chat room) is used where how meeting will be conducted.

•The goal of this meeting is to identify the problem propose elements of the solution, negotiate different approaches and specify a preliminary set of solution requirement in an atmosphere for the achievement of the goal.

Requirements elicitation Techniues:

Department of Computer Science and Engineering

5. Quality Function Deployment:

In this technique customer satisfaction is of prime concern, hence it emphasizes on the requirements which are valuable to the customer.

It is used to define Quality function

Itimprove uality of software.

Requirements elicitation Techniues:

3 types of requirements are identified in QFD –

1. **Normal requirements –**

In this the **objective and goals** of the proposed software are discussed with the customer. Example – normal requirements for a result management system may be entry of marks, calculation of results, etc. Easily identify during the meeting with costUmer.

Example;- Mouse handling,keyboard event

Requirements elicitation Techniues:

2. Expected requirements –

These requirements are so obvious that the customer need not explicitly state them.

Example – protection from unauthorized access.

3. Exciting requirements –

It includes features that are beyond customer's expectations and prove to be very satisfying when present. Example – when unauthorized access is detected, it should backup and shutdown all processes.

Example:- Spell check facility.

Requirements elicitation Techniues:

4. Function Deployment –

Value of each function

5. information Deployment –

Data object identified

6. Task Deployment –

Task with each function identified

7. Value analysis –

Priority of requirement

Requirements elicitation Techniues:

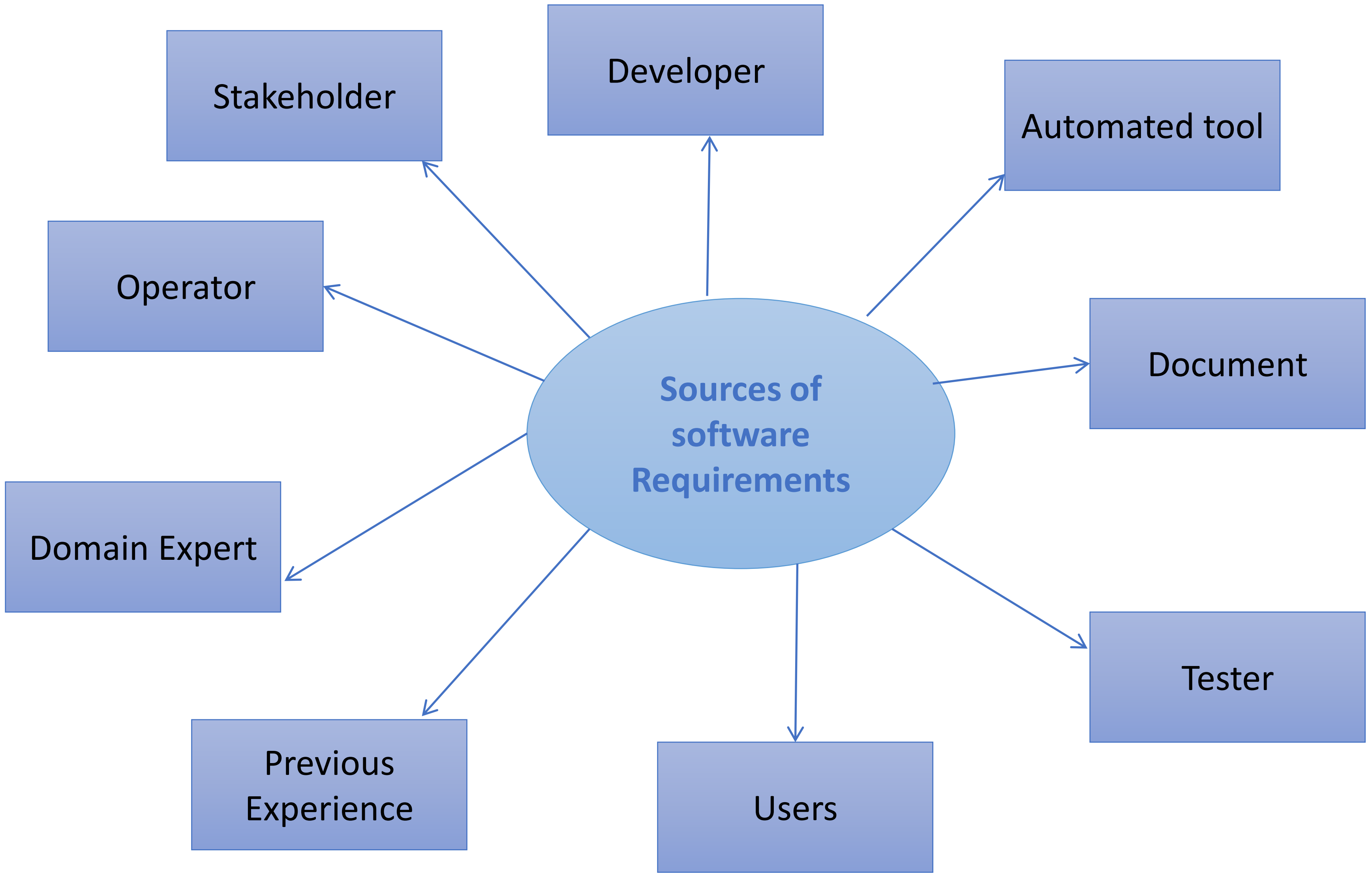
6. **Domain analysis:-** Skilled expertize with specific Area

7. **Observations:** By seeing workflow of organization or business environment

8. **Prototyping:-** Existing requirement framework,

9. **Use Case Approach:-** Different function of software.

Sources of Requirements



Sources of Requirements

1. **Stakeholders/Buyers :**

They are the persons responsible for accepting and executing the software. They can be individual individuals, organizations, trusts or even the government or public of a country.

2. **User/Beneficiaries :**

These are the users of the product for which the product is intended.

Sources of Requirements

3. **Operators :**

They are the persons who work on the software to make the services of the software available to its beneficiaries or the end users.

4. **Developer :**

The software engineering responsible for developing the software to make it provide the expected services. They are responsible for software design, prototype development, and technical feasibility. They work closely with the end-users, buyers, and application experts

Sources of Requirements

5. Domain experts :

They are professionals with experience and expertise of the domain in which the software provides its services, viz. insurance, financials, banking, communication, data transfer, networking, etc. Domain experts unwind the hidden or unseen probable requirements or risks involved in product development.

Sources of Requirements

6. Automated Tools :

In the new generation of information technology and software development paradigm, many automated and semi-automated tools are available that allow for the affirmation and management of the need for building the system. such software also provides input. System/software requirements.

Sources of Requirements

7. Past Experience/Case Studies :

An organization working in the similar or same domain may provide its past experience or even documented case studies. This helps have a clearer picture of the requirements, which may otherwise be left hidden.

Sources of Requirements

8. Connected People/Machine/Environment :

People associated with software or environmental factors and IT domain may give a lot of provide information about constraints involved in development, development, its and environment implications on software.

Sources of Requirements

9. Tester :

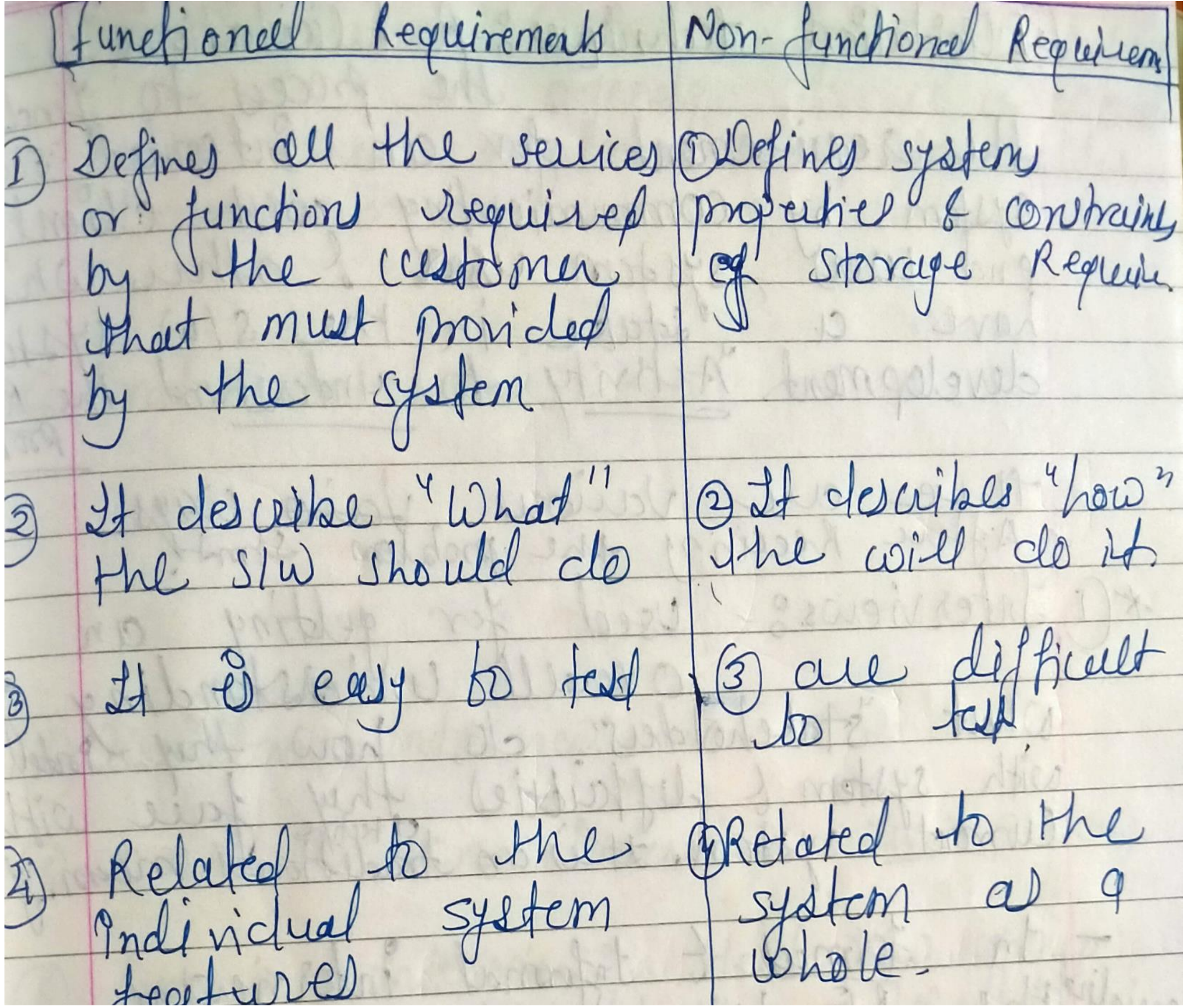
Testers are a good source of information about the user’s behavior or the predictive behavior of the system’s condition. continuous contact with real users for their input. In such cases, examiners may use their experiences and analytical skills to provide input.

FUNCTIONAL REQUIREMENTS

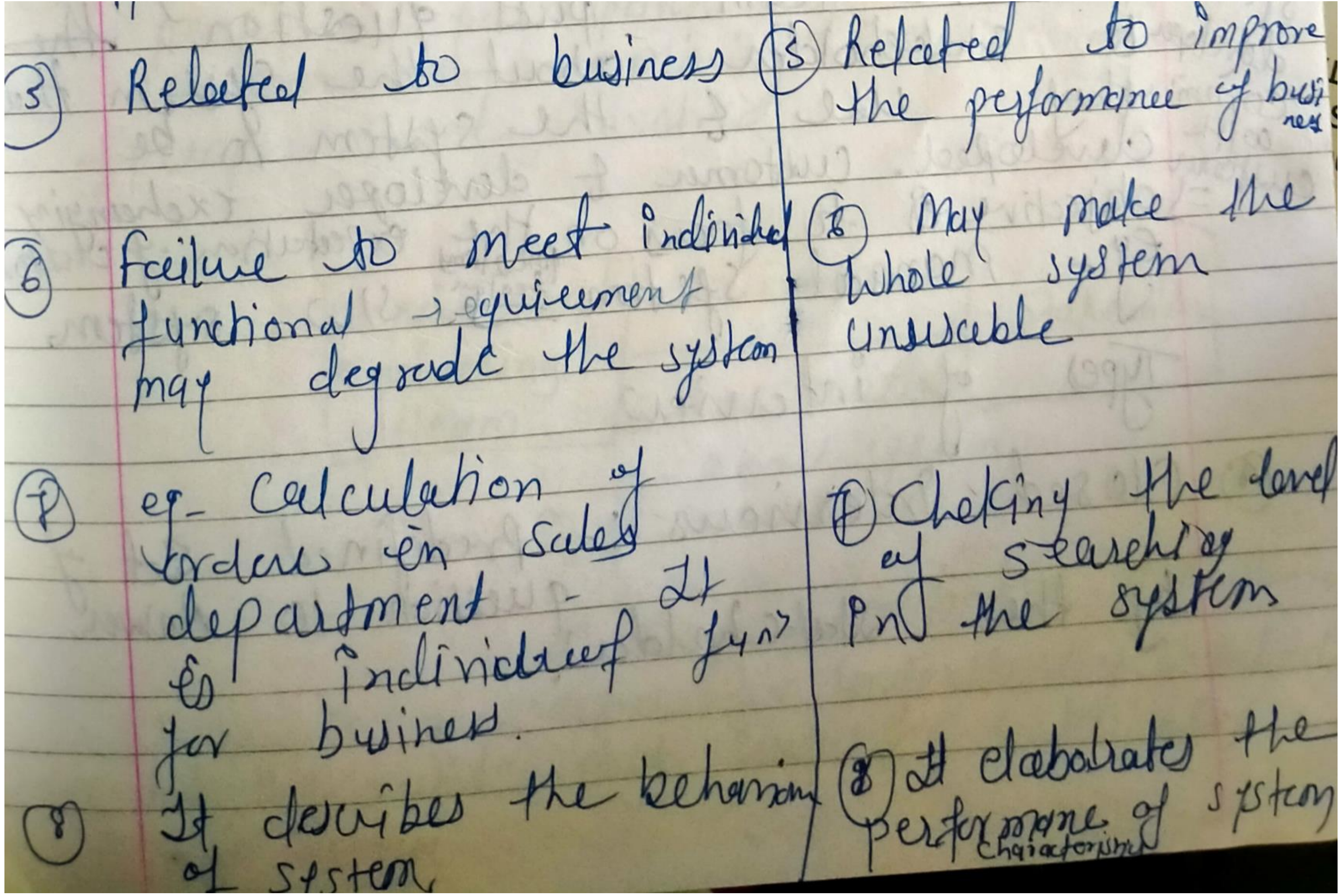
- It is also called behavioural reuirements.
- Describe functionalities or services that should providedby software.
- Specifies:- Inputs,Outputs ,external interface& thE functionas that should not be included in software.
- Desribe Procedure
- **Example online banking**

NON - FUNCTIONAL REQUIREMENTS

- It is also called Quality reuirements.
- Describe Attribus of system
- Specifies:- User requirement,budget Constraints & org policies,
- Not related directly to system.
- TYPES:- 1. Product 2. Organizational 3. External
- **Example :- IF aeroplane is unalbe to fullfill reliAbiliy requirement .it is nor approved for safe operationS.**



Functional Requirements	Non-functional Requirem
1 Defines all the services or functions required by the customer that must provided by the system	1 Defines system properties & constraints of storage Requir
2 It describe "What" the S/W should do	2 It describes "how" the will do it.
3 It is easy to test	3 are difficult to test.
4 Related to the Individual system features	4 Related to the system as a whole.



3) Releated to business	5) Related to improve the performance of business
6) Failure to meet individual functional requirement may degrade the system	6) May make the whole system unsuable
7) eg- Calculation of order in sales department - It is individual fun^n for business.	7) Cheking the level of searching in the system
8) It describes the behaviour of system	8) It elaborates the performance of system characteristics

9. Behavioural Requirements	9. Quality Requirements
10.**Example** online banking services	10. **Example** Aeroplane not Reliable

Analysis Modeling

1. It is a technical representation of the system. Example : home
2. The software engineer /Analyst builds the model using requirement elicited from customer.
3. It consist with static and dynamic models
4. Model is a simplification of reality
5. It uses a combination of text and diagrams.
6. Before develop a software ,we understand a software easily.
7. By building analysis models, it becomes easy to uncover requirement inconsistencies & omission.

Analysis Modeling Principles

1. The information domain of a problem must be represented and understood.

2. The functions that are software performs must be defined.

3. The behavior of the software must be represented.

4. The models that depict information, function and behavior must be partitioned in a manner that uncovers detail in a layered fashion.

5. The analysis task should move from essential information toward implementation detail.

6. it describe what the customer require.

7. Establish basis for creation of software.

Analysis Modeling Principles

1. The information domain of a problem must be represented and understood.

2. The functions that are software performs must be defined.

3. The behavior of the software must be represented.

4. The models that depict information, function and behavior must be partitioned in a manner that uncovers detail in a layered fashion.

5. The analysis task should move from essential information toward implementation detail.

Analysis Modeling Principles

1. Information domain encompasses that the data flow into the system, out of the system and data stored.

2. Functions provide direct benefit to end-users and also provide internal support for those features that are user visible.

3. Behavior driven by its interaction with the external environment. E.g. Input provided by end-users, control data provided by an external system, or monitoring data.

Analysis Modeling Principles

4. Key strategy of analysis model, divide complex problem into sub-problem until each sub-problem is relatively easy to understood. This concept is called partitioning.

5. The “essence” of the problem is described without any consideration of how a solution will be implemented. E.g. Video game requires that player “instruct” Implementation detail (design model) indicates how the essence will be implemented E.g. Keyboard command might be typed or a joystick used.

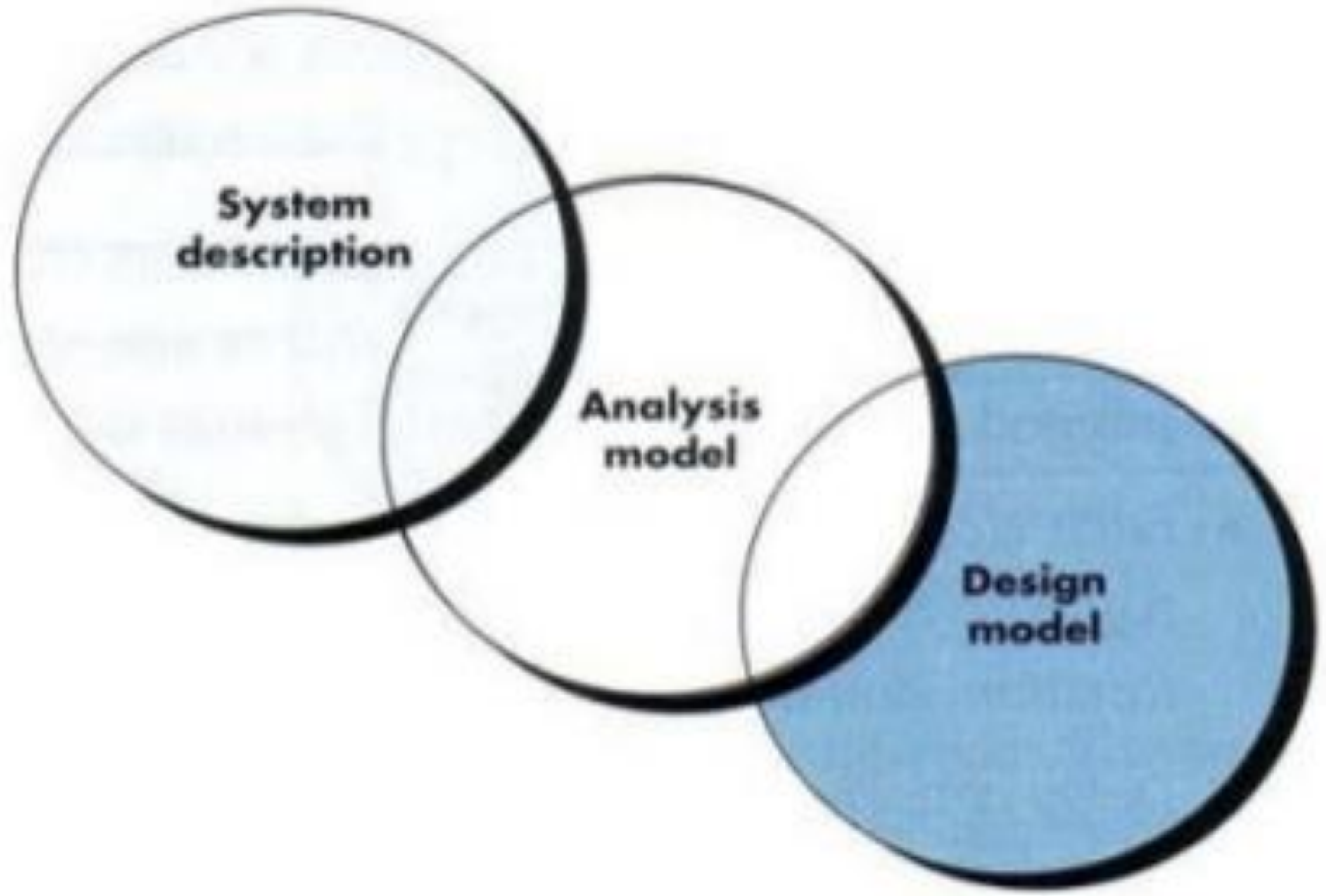
Objective :-Analysis Model AS Connector

Three Primary Objectives:

1. Describe what the customer requires.
2. Establish a basis for the creation of a software design.
3. Devise a set of valid requirements that can be validated once the software is built.

Its bridges the gap between a system-level description that describes overall system functionality and a software design.

Analysis Model - A Bridge



Analysis Model Objectives

- ❖ Team choose one approach & makes the representation from it.

[Best from both]

- ❖ Model representation help in building S/w design

Analysis Model Objectives

Guidelines :

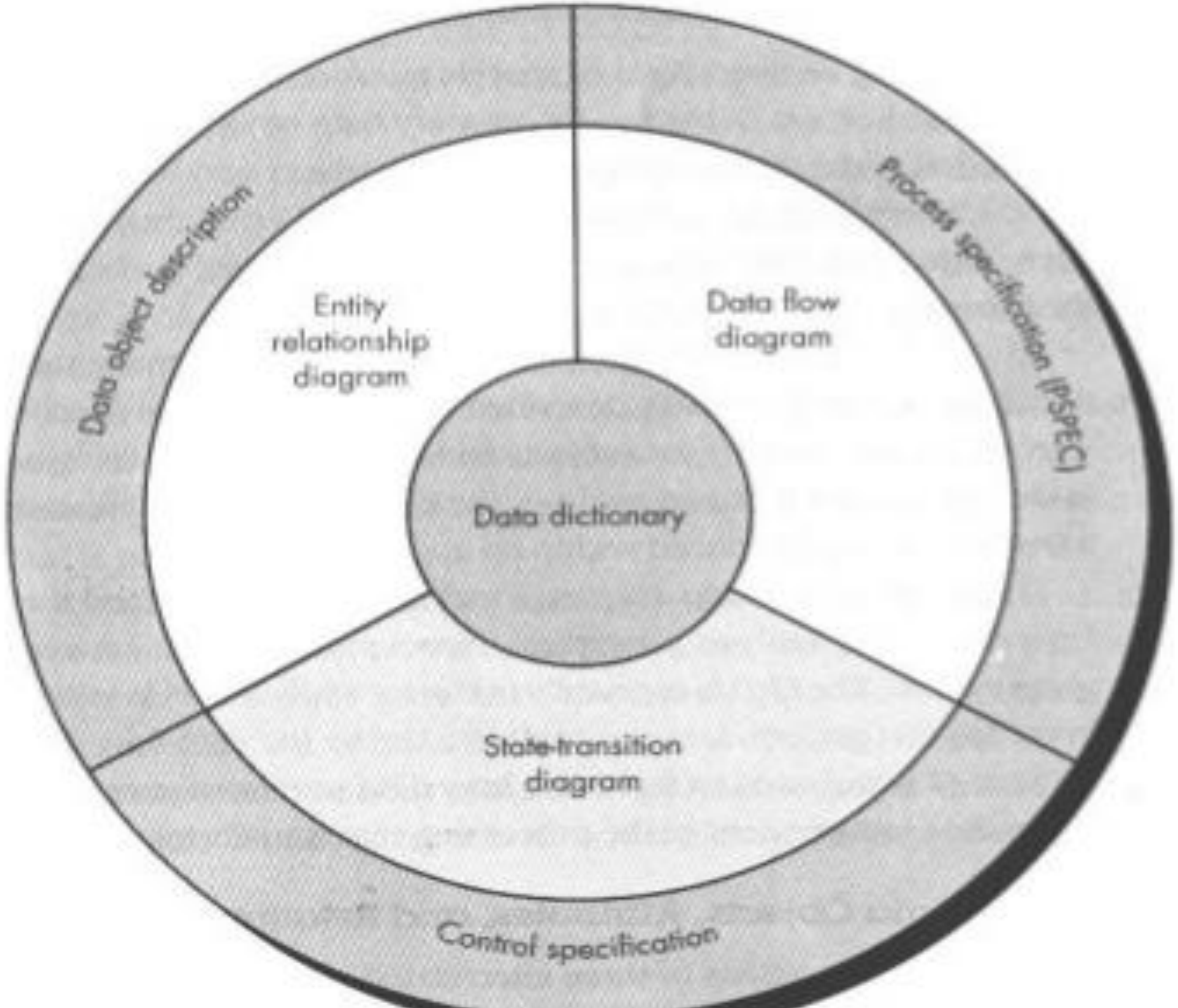
- ❖ Graphics should be used whenever possible.
- ❖ Differentiate between the logical (essential) and physical (implementation) considerations.
- ❖ Develop a way to track and evaluate user interfaces.

Elements/ Structure of Analysis Model

The analysis model must achieve the primary objective:

- It must describe the requirements of a customer,
- It must establish a basis for the creation of a software design ,and
- It must define a set of requirements, which can be validated after the complete software is built.

Elements/ Structure of Analysis Model



Elements/ Structure of Analysis Model

•**Data Object Description:**

It stores and provides complete knowledge about a data object present and used in the software. It also gives us the details of attributes of the data object present in the Entity Relationship Diagram. Hence, it incorporates all the data objects and their attributes.

Elements/ Structure of Analysis Model

•**Process Specification:**

It stores the description of each function present in the data flow diagram. It describes the input to a function, the algorithm that is applied for the transformation of input, and the output that is produced. It also shows regulations and barriers imposed on the performance characteristics that are applicable to the process and layout constraints that could influence the way in which the process will be implemented.

Elements/ Structure of Analysis Model

•**Control Specification:**

It stores additional information about the control aspects of the software. It is used to indicate how the software behaves when an event occurs and which processes are invoked due to the occurrence of the event. It also provides the details of the processes which are executed to manage events.

Elements/ Structure of Analysis Model

- **Data Dictionary:**

It is a repository that consists of a description of all data objects used or produced by the software. It stores the collection of data present in the software. It is a very crucial element of the analysis model. It acts as a centralized repository and also helps in modeling data objects defined during software requirements.

Elements/ Structure of Analysis Model

•**State Transition Diagram:** It shows various modes of behavior (states) of the system and also shows the transitions from one state to another state in the system. It also provides the details of how the system behaves due to the consequences of external events. It represents the behavior of a system by presenting its states and the events that cause the system to change state. It also describes what actions are taken due to the occurrence of a particular event.

Elements/ Structure of Analysis Model

- **Data Flow Diagram (DFD):** It depicts the functions that transform data flow and it also shows how data is transformed when moving from input to output. It provides the additional information which is used during the analysis of the information domain and serves as a basis for the modeling of function. It also enables the engineer to develop models of functional and information domains at the same time.

Elements/ Structure of Analysis Model

- **Entity Relationship Diagram (ERD):** It depicts the relationship between data objects and is used in conducting data modeling activities. The attributes of each object in the Entity-Relationship Diagram can be described using Data object description. It provides the basis for activity related to data design.

Purpose of SRS document?

- SRS establishes basis of agreement between the user and the supplier.
  - Users needs 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 communications gap, and
  - specifies user needs in a manner both can understand

Need for SRS…

- Software requirments Provides basis for creating the SRS.

- Helps user understand his needs.
  - users do not always know their needs
  - must analyze and understand the potential
  - The requirement 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 “

Need for SRS…

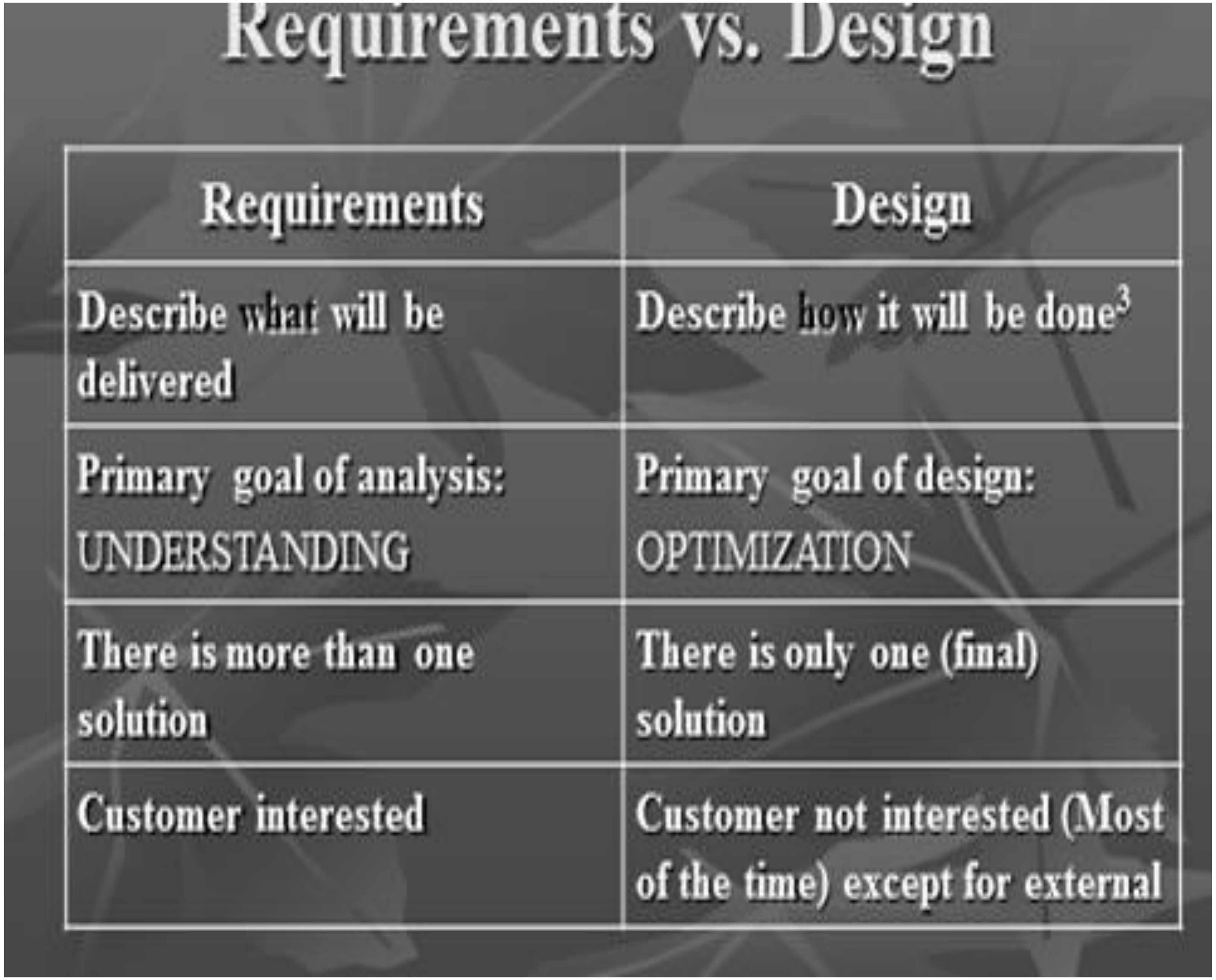
- High quality SRS essential for high Quality SW
  - Requirement errors get manifested in final sw
  - To satisfy the quality objective, must begin with high quality SRS

  - Requirements defects cause later problems
    - 25% of all defects in one study; 54% of all defects found after user testing
    - defects often found in previously approved SRS.
    - Useful in plan, cost performance tasks,tracking,progressof develomnent.

Need for 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

Department of Computer Science and Engineering



Requirements vs. Design

Requirements	Design
Describe what will be delivered	Describe how it will be done[3]
Primary goal of analysis: UNDERSTANDING	Primary goal of design: OPTIMIZATION
There is more than one solution	There is only one (final) solution
Customer interested	Customer not interested (Most of the time) except for external

Requirements Documentation

SRS Should

-- Correctly define all requirements

-- not describe any design details

-- not impose any additional constraints

Characteristics of a good SRS

An SRS Should be

✓ Correct

✓ Unambiguous

✓ Complete

✓ Consistent

Requirements Documentation

- ✓ Ranked for important and/or stability
- ✓ Verifiable
- ✓ Modifiable
- ✓ Traceable

Requirements Documentation

Correct

An SRS is correct if and only if every requirement stated therein is one that the software shall meet.

Unambiguous

An SRS is unambiguous if and only if, every requirement stated therein has only one interpretation.

Complete

An SRS is complete if and only if, it includes the following elements

(i) All significant requirements, whether related to functionality, performance, design constraints, attributes or external interfaces.

Requirements Documentation

(ii) Responses to both valid & invalid inputs.

(iii) Full Label and references to all figures, tables and diagrams in the SRS and definition of all terms and units of measure.

Consistent

An SRS is consistent if and only if, no subset of individual requirements described in it conflict.

Ranked for importance and/or Stability

If an identifier is attached to every requirement to indicate either the importance or stability of that particular requirement.

Requirements Documentation

Verifiable

An SRS is verifiable, if and only if, every requirement stated therein is verifiable.

Modifiable

An SRS is modifiable, if and only if, its structure and style are such that any changes to the requirements can be made easily, completely, and consistently while retaining structure and style.

Traceable

An SRS is traceable, if the origin of each of the requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation.

Organization of SRS

1. Introduction	
1.1	Purpose
1.2	Scope
1.3	Definition, Acronyms and abbreviations
1.4	References
1.5	Overview

Requirements Documentation

2. The Overall Description

2.1 Product Perspective

2.1.1 System Interfaces

2.1.2 Interfaces

2.1.3 Hardware Interfaces

2.1.4 Software Interfaces

2.1.5 Communication Interfaces

2.1.6 Memory Constraints

2.1.7 Operations

2.1.8 Site Adaptation Requirements

Requirements Documentation

2.2 Product Functions
2.3 User Characteristics
2.4 Constraints
2.5 Assumptions for dependencies
2.6 Apportioning of requirements

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 attributes
   3.7 Organization of specific requirements
   3.8 Additional Comments.

4. Others Requirements/Information

Introduction

- The Software Requirement Specification(SRS) document provides an overview of the entire SRS.
  - **Purpose:**
    - Identify the purpose of SRS and its intended audience**.**
  - **Scope:**
    - Identify the software products to be produced by name.
    - Explain what the software products will, and, if necessary, will not do.
    - Describe the application of the software being specified, including relevant benefits, objective and goals
    - Be consistent with similar statement in higher level specification if the exist.

Introduction CONTD……………….

- **Definition, Acronyms, and Abbreviations:**
    - Provide definition of all terms, acronyms, and abbreviations required to properly interpret the SRS.
    - This information may be provided by reference to an appendix.

- **References:**
    - Provide complete list of all documents referenced elsewhere in the SRS.
    - Identified each document by title, report number, date, and publishing organization.
    - Specify the sources from which the reference can be obtained.

Introduction CONTD…………………

- **Overview:**
  - Describe what the rest of the SRS contains.
  - Explain how the SRS is organised.

The overall description

- Describe the general factors that affects the product and its requirements.
- **Product Perspective:**
  - Put the product into perspective with other related products.
  - If the product is independent and totally self-contained, it should be so stated here.
  - If the SRS defines a product that is a component of a larger system, as frequently occurs, then it relates the requirements of the larger system to functionality of the software and identifies interfaces between that system and the software**.**

The overall description contd……

The following subsection describe how the software operates inside various constraints:

- **System Interface:**
  - List each system interface and identify the functionality of the software to accomplish the system requirement.
  - And the interface description to match the system.

The overall description contd……

- **Interfaces:**
  - It specify the logical characteristic of each interface between the software product and its users.
  - All the aspects of optimising the interface with the person who must use the system.
- **Hardware Interfaces:**
  - Specify the logical characteristics of each interface between the software product and hardware component of the system.
  - This also includes configuration characteristics.

The overall description contd……

- **Software interfaces:**
  - Specify the use of other software products and interface with other application system.
  - It includes
    - Name
    - Specification number
    - Version number
    - Source

The overall description contd……

- **Communication Interfaces:**
    - Specify the various interfaces to communications such as local network protocols, etc.
- **Memory constraints:**
    - Specify any applicable characteristics and limits on primary and secondary memory.

The overall description contd……

- **Operations:**
  - Specify the normal and special operations required by the users such as:
    - The various mode of operation in the user organization.
    - Periods of interactive operations and periods of unattended operations.
    - Data processing support functions
    - Backup and recovery operations

The overall description contd……

- **Site Adaption Requirements:**
  - Define the requirement for any data or initialization sequences that are specific to a given site, mission, operational mode.
  - Specify the site or mission related feature that should be modified to adapt the software to a particular installation.

Product function

- Provide a summary of the major functions that the software will perform.
  - The function should be organised in a way that makes the list of functions understandable to the customer or to any one else reading the document for the first time.
  - Textual or graphic methods can be used to show the different function and their relationship.

User characteristics

- Describe those general characteristic of the intended users of the product including educational level, experience, and technical expertise

constraints

- Provide a general description of any other items that will limit the developer’s options. These can include:
  - Regulatory Policies
  - Hardware Limitation
  - Interface to other Application
  - Parallel Operation
  - Audit Function
  - Control Function
  - Higher Order Language Requirements
  - Signal Handshake Protocols
  - Reliability Requirements
  - Criticality of the Application
  - Safety and Security Considerations

Assumption and dependencies

- List each of the factors that affects the requirements stated in SRS
- These factors are not designed constraints on the software but, any change to them may that can affect requirements in SRS.

Apportioning of requirements

- Identify requirements that may be delayed until future version of the system.

Specific Requirements

- Specific requirements should be stated with all the characteristics of the good SRS.
  - Correct
  - Unambiguous
  - Complete
  - Consistent
  - Ranked for importance and stability
  - Verifiable
  - Modifiable
  - Traceable
- Specific requirement should be cross-referenced to earlier documents that relate.
- All requirements should uniquely identifiable.
- Careful attention should be given to organising the requirement to maximize readability.

External interface

- This contains a detailed description of all the input into and output from the software system.
- It contains both content and format as follow:
  - Name of item
  - Description of purpose
  - Source of input or destination of output
  - Valid range, accuracy and tolerance
  - Units of measure
  - Timing
  - Relationship to other Input/Outputs
  - Screen formats
  - Window formats
  - Data formats
  - Command formats
  - End message

Functions

- Functional requirements defines the fundamental action that must take place in software in accepting and processing the inputs and in processing and generating the outputs. These includes:
  - Validity checks on the inputs
  - Exact sequence of operations
  - Response to abnormal situation, including
    - Overflow
    - Communication facilities
    - Error handling and recovery
  - Effects of parameters
  - Relationship of output to inputs, including
    - Input/output Sequences
    - Formulas for input to output conversion.

Performance Requirements

- This subsection specifies both the static and the dynamic numerical requirements placed on the software.
- Static numerical requirements may include:
  - The number of terminals to be supported.
  - The number of simultaneous users to be supported.
  - Amount and types of information to be handled.
- Dynamic numerical requirements may include:
  - Number of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

Logical database requirements

- This section specifies the logical requirements for any information that is to be placed into a database. This may include:
  - Types of information used by various functions
  - Frequency of use
  - Accessing capabilities
  - Data entities and their relationship
  - Integrity constraints.

Design constraints

- Specify design constraints that can be imposed by other standards, hardware limitations, etc.
- **Standard Compliance:**
  - Specify the requirements derived from existing standards or regulations.
  - Report format
  - Data naming
  - Accounting procedure
  - Audit tracing

Software system attributes

- There are a number of quality attribute of software that can serve as requirements.
- **Reliability:**
  - Specify the factor required to establish the required reliability of the software system at time of delivery.
- **Availability:**
  - Specify the factors required to guarantee a defined availability level for the entire system such as check point, recovery and restart.

Software system attributes contd……

- **Security:**
  - Specify the factor that would protect the software form accidental or malicious access, use, modification, destruction. This may include need to:
    - Utilizing certain cryptographic techniques
    - Keep specific lo or history data sets
    - Assign certain functions to different modules
    - Restrict communication between some areas of the program
    - Check data integrity for critical variable

Software system attributes contd……

- **Maintainability:**
  - Specify attribute of software that relate to the ease of maintenance of software itself.
  - There may be some requirement for certain modularity, interface, complexity, etc.
- **Portability:**
  - Specify attributes of software that relates to the ease of parting the software to other host machine and operating systems. This may include:
    - Percentage of component with host dependent code
    - Percentage of code that is host dependent
    - Use of a proven portable language
    - Use of a particular compiler
    - Use of a particular operating system.

Organising the specific requirements

- **System Mode:**
  - Some system behave quite differently depending on the mode of operation. There are two possible outlines.
  - The choice depends on whether interfaces,
  - Performance are dependent on mode.
- **User Class:**
  - Some system provide different sets of functions to different class of users.
- **Objects:**
  - Object are real world entities that have a counterpart within the system.
  - Associated with each object is a set of attributes and functions.
  - These functions are called services, methods, or processes.

Organising the specific requirements contd…

- **Feature:**
  - A feature is an externally desired service by the system that may require a sequence of inputs to effect the desired result.
  - Each feature is described as sequence of stimulus-response pairs.
- **Stimulus:**
  - Some system can be best organised by describing their functions in terms of stimuli.
- **Response:**
  - Some system can be best organised by describing their function in support of the generation of the response.

Change Management Process:

- Specify the change management process to be used to identify, log, evaluate, and update the SRS to reflect changes in project scope and requirement.

Document approval

- Identify the approvers of the SRS documents. Approver’s name, signature, and date should be used.

Supporting information

- The supporting information makes the SRS easier to use. It includes:
  - Table of contents
  - Index
  - Appendices

SRS: Example

- **1.0 INTRODUCTION**

This document specifies all 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 …..

SRS

- **1.3 Definitions, Acronyms, and Abbreviations**

SRS - Software Requirements Specifications

IEEE - Institute of Electrical and Electronic Engineering

- **1.4 Reference**

[1] IEEE 830-1993: IEEE Recommended Practice for Software Requirements Specifications" IEEE Standards Collection, IEEE, 1997.

- **1.5 Overview**

In the following sections of this specification……will be presented.

In Section 2, the general product and its functions will be introduced.

SRS

In Section 3, all detailed requirements will be specified and grouped.

In Appendix …….

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

- **2.3 User Characteristics**

The users of the system are:

- ❿**Level of Users' Computer Knowledge**
- ❿**Level of Users' Business Knowledge**
- ❿**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 ……

SRS

- **SPECIFIC REQUIREMENTS**
  3.1 Functional Requirements
  3.1.1 Unit Registration
- The unit registration requirements are concerned with functions regarding unit registration which includes students selecting, adding, dropping, and changing a unit.
- **SRS-001 (3.1.1.1):**
  - The system shall allow the user to register a unit.
- **SRS-002 (3.1.1.2):**
  - System shall allow the user to delete a unit if the user has chosen to drop that unit.
- **SRS-003 (3.1.1.3):**
  - System shall check if a unit has been filled by enough registered students.

SRS

- **SRS-004 (3.1.1.4):**
- System shall allow the user to add his/her name to the unit waiting list if the user wants to register in a unit which has been filled already with enough registered students.
- **SRS-005 (3.1.1.5):**
- System shall automatically register the unit for the user who is the first one on the waiting list if a vacancy appears for that unit.
- **SRS-006 (3.1.1.6):**
- System shall allow the user to change practical session(s) within a unit.
- **SRS-007 (3.1.1.7):**
- System shall allow the user to change tutorial session(s) within a unit.

SRS

- **3.1.2 Retrieving and Displaying Unit Information**
- The retrieving and displaying requirements are concerned with how information is retrieved and presented to the user.
- **SRS-014 (3.1.2.1):**
- The system shall allow users to enter the following selection criteria to retrieve unit information: by unit code, by unit number, by title of unit, by weight of unit (credit points).
- **OR** by unit code **(3.1.2.1.1)** , by unit number **(3.1.2.1.2)** , by title of unit **(3.1.2.1.3)** , by weight of unit (credit points) **(3.1.2.1.4).**

SRS

- **3.2 Design Constraints**
- **SRS-031 (3.2.1):**
- System shall store and retrieve persistent data.
- **SRS-032 (3.2.2):**
- System shall support PC and/or UNIX platforms.
- **SRS-033 (3.2.3):**
- System shall be developed using the JAVA programming language

SRS

- **3.3 Non-Functional Requirements**
- **SRS-034 (3.3.1):**
- System shall respond to any retrieval in less than 5 seconds.
- **SRS-035 (3.3.2):**
- System shall generate a report within 1 minute.
- **SRS-036 (3.3.3):**
- System shall allow the user to remotely connect to the system.
- **SRS-041 (3.3.8):**
- The system will be accompanied by a comprehensive user manual.

SRS

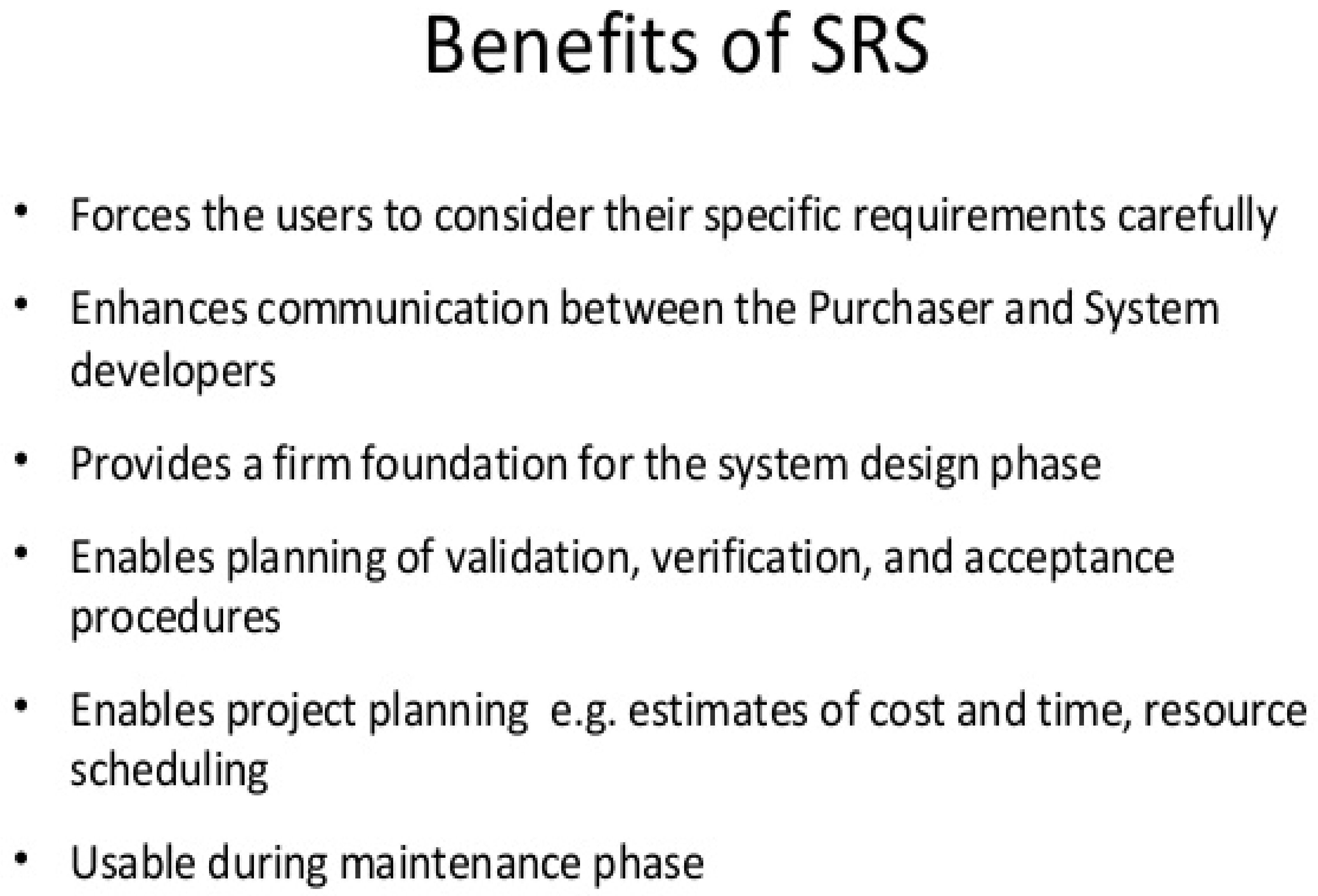
- **3.5.3 Security**
- The security requirements are concerned with security and privacy issues.

SRS-029:

- System shall provide staff ID and password verification protection to protect from unauthorised use of the system.

SRS-030:

- System shall allow the store manager to add, remove and modify staff ID and passwords as required.



Benefits of SRS

- Forces the users to consider their specific requirements carefully
- Enhances communication between the Purchaser and System developers
- Provides a firm foundation for the system design phase
- Enables planning of validation, verification, and acceptance procedures
- Enables project planning e.g. estimates of cost and time, resource scheduling
- Usable during maintenance phase

SRS Review

- Formal Review done by Users, Developers, Managers, Operations personnel
- To verify that SRS confirms to the actual user requirements
- To detect defects early and correct them.
- Review typically done using checklists.

Requirement Validation

- It is a process of checking that requirements defined for developments & the customer really wants.
- It check error at initial phase of development an error as increase excessive reworks.
- We perform different types of test and checking.
- It is a technique in which the description in SRS document is verified for system implementation.
- it'S objective is to certify the SRS document.
- To check issues related to requirements
- It works with final draft of SRS document.

Validation objectives

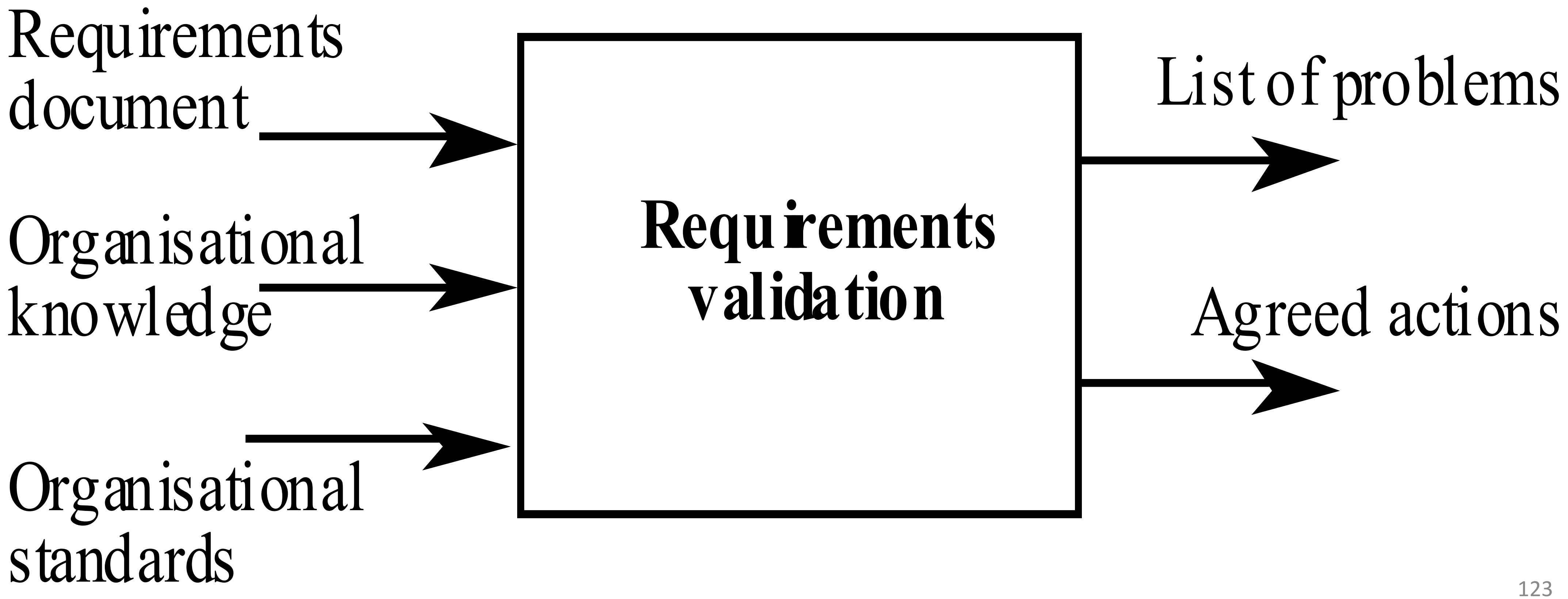
- Certifies that the requirements document is an acceptable description of the system to be implemented
- Checks a requirements document(SRS) for
  - ✓Completeness and consistency
  - ✓Conformance to standards(Validity checks)
  - ✓Requirements conflicts
  - ✓Technical errors
  - ✓Ambiguous requirements
  - ✓Realism checks

Analysis and validation

- Analysis works with raw requirements as elicited from the system stakeholders
  - “Have we got the right requirements” is the key question to be answered at this stage
- Validation works with a final draft of the requirements document i.e. with negotiated and agreed requirements
  - “Have we got the requirements right” is the key question to be answered at this stage

Requirement Validation: inputs and outputs

Requirement document:-	SRS documnet
Organisational knowledge:-	Judge the realism
Organisational standard:-	Verified



Validation inputs

1. **Requirements document**
   - Should be a complete version of the document, not an unfinished draft. Formatted and organised according to organisational standards. This contains the SRS of Software.
2. **Organisational knowledge**
   - Knowledge, often implicit, of the organisation which may be used to judge the realism of the requirements
3. **Organisational standards**
   - Local standards e.g. for the organisation of the requirements document(SRS) .Standards verified during validation.

Validation outputs

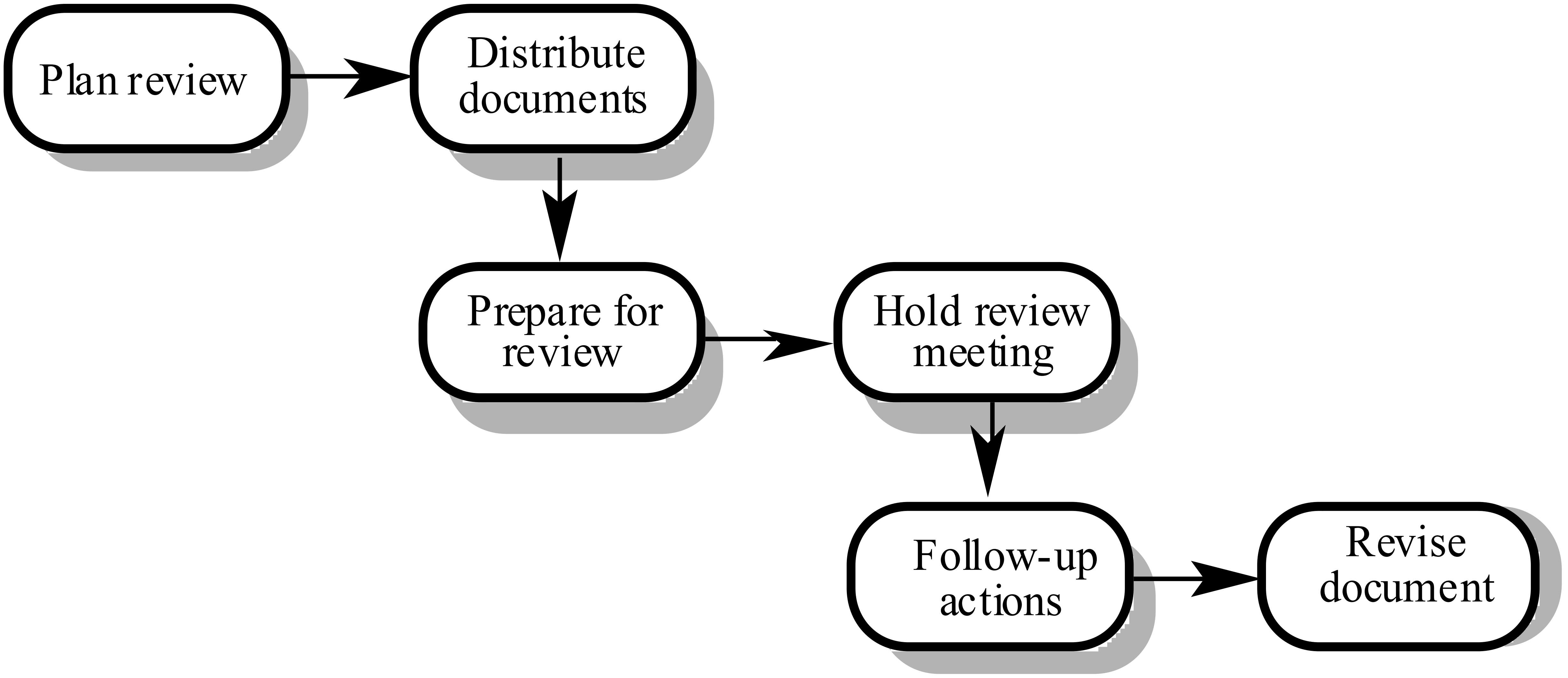
- **Problem list**
  - List of discovered problems in the requirements document
- **Agreed actions**
  - List of agreed actions in response to requirements problems. Some problems may have several corrective actions; some problems may have no associated actions

Requirement Validation Techniue

Requirements reviews

- A group of people read and analyse the requirements, look for problems, meet and discuss the problems and agree on actions to address these problems

Requirements review process



Review activities

- **Plan review**
  - The review team is selected and a time and place for the review meeting is chosen.
- **Distribute documents**
  - The requirements document is distributed to the review team members
- **Prepare for review**
  - Individual reviewers read the requirements to find conflicts, omissions, inconsistencies, deviations from standards and other problems.

Review activities

- **Hold review meeting**
  - Individual comments and problems are discussed and a set of actions to address the problems is agreed.
- **Follow-up actions**
  - The chair of the review checks that the agreed actions have been carried out.
- **Revise document**
  - The requirements document is revised to reflect the agreed actions. At this stage, it may be accepted or it may be re-reviewed

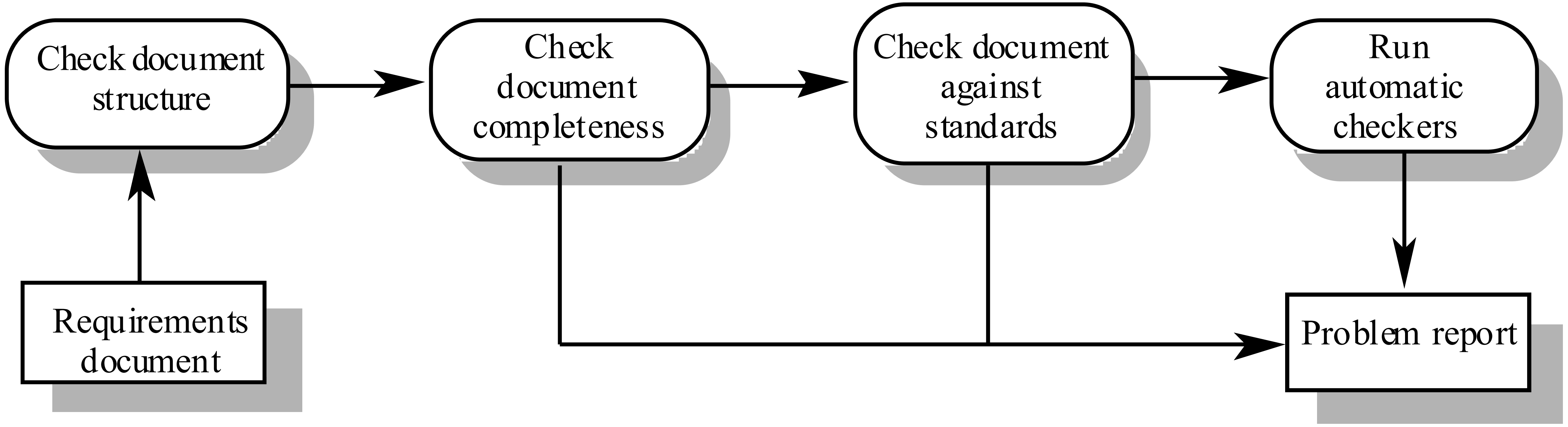
Problem actions

- Requirements clarification
  - The requirement may be badly expressed or may have accidentally omitted information which has been collected during requirements elicitation.
- Missing information
  - Some information is missing from the requirements document. It is the responsibility of the requirements engineers who are revising the document to discover this information from system stakeholders.
- Requirements conflict
  - There is a significant conflict between requirements. The stakeholders involved must negotiate to resolve the conflict.
- Unrealistic requirement
  - The requirement does not appear to be implementable with the technology available or given other constraints on the system. Stakeholders must be consulted to decide how to make the requirement more realistic.

Pre-review checking

- Reviews are expensive because they involve a number of people spending time reading and checking the requirements document
- This expense can be reduced by using pre-review checking where one person checks the document and looks for straightforward problems such as missing requirements, lack of conformance to standards, typographical errors, etc.
- Document may be returned for correction or the list of problems distributed to other reviewers

Pre-review checking



Review team membership

- Reviews should involve a number of stakeholders drawn from different backgrounds
    - People from different backgrounds bring different skills and knowledge to the review
    - Stakeholders feel involved in the RE process and develop an understanding of the needs of other stakeholders
- Review team should always involve at least a domain expert and an end-user

Review checklists

- **Understandability**
  - Can readers of the document understand what the requirements mean?
- **Redundancy**
  - Is information unnecessarily repeated in the requirements document?
- **Completeness**
  - Does the checker know of any missing requirements or is there any information missing from individual requirement descriptions?
- **Ambiguity**
  - Are the requirements expressed using terms which are clearly defined? Could readers from different backgrounds make different interpretations of the requirements?

Review checklists

- **Consistency**
  - Do the descriptions of different requirements include contradictions? Are there contradictions between individual requirements and overall system requirements?
- **Organisation**
  - Is the document structured in a sensible way? Are the descriptions of requirements organised so that related requirements are grouped?
- **Conformance to standards**
  - Does the requirements document and individual requirements conform to defined standards? Are departures from the standards, justified?
- **Traceability**
  - Are requirements unambiguously identified, include links to related requirements and to the reasons why these requirements have been included?

Checklist questions

- Is each requirement uniquely identified?
- Are specialised terms defined in the glossary
- Does a requirement stand on its own or do you have to examine other requirements to understand what it means?
- Do individual requirements use the terms consistently
- Is the same service requested in different requirements? Are there any contradictions in these requests?
- If a requirement makes reference to some other facilities, are these described elsewhere in the document?
- Are related requirements grouped together? If not, do they refer to each other?

Requirements problem example

- “4. EDDIS will be configurable so that it will comply with the requirements of all UK and (where relevant) international copyright legislation.
- Minimally, this means that EDDIS must provide a form for the user to sign the Copyright Declaration statement.
- It also means that EDDIS must keep track of Copyright Declaration statements which have been signed/not-signed.
- Under no circumstances must an order be sent to the supplier if the copyright statement has not been signed.”

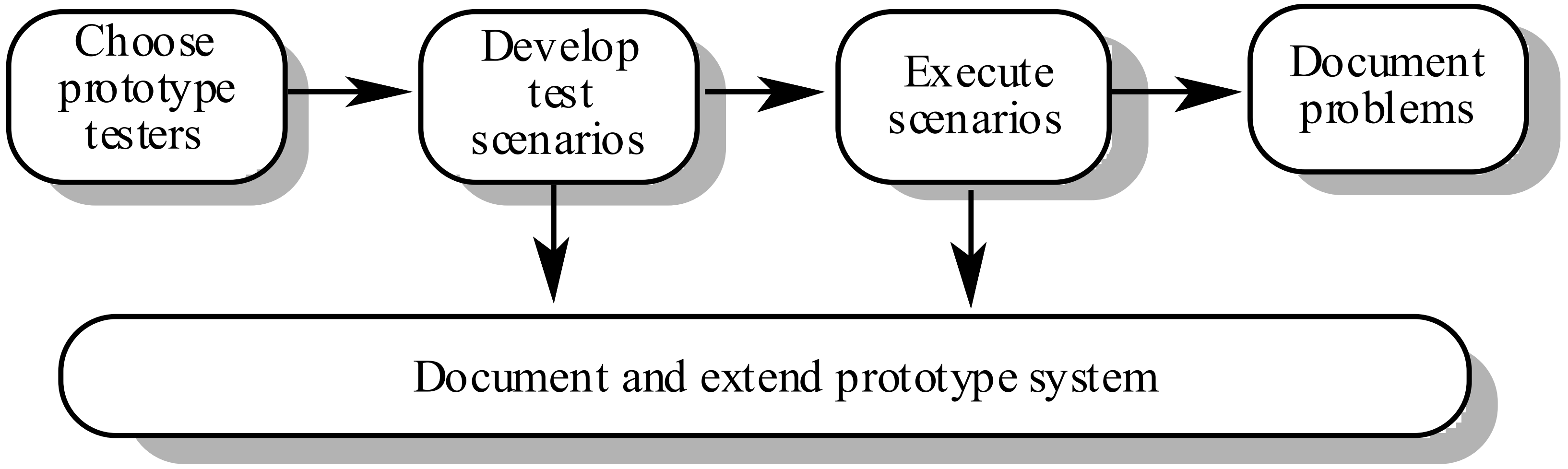
Problems

- Incompleteness
  - What international copyright legislation is relevant?
  - What happens if the copyright declaration is not signed?
  - If a signature is a digital signature, how is it assigned?
- Ambiguity
  - What does signing an electronic form mean? Is this a physical signature or a digital signature?
- Standards
  - More than 1 requirement. Maintenance of copyright is one requirement; issue of documents is another

2. Prototyping

- Prototypes for requirements validation demonstrate the requirements and help stakeholders discover problems
- Validation prototypes should be complete, reasonably efficient and robust. It should be possible to use them in the same way as the required system
- User documentation and training should be provided

Prototyping for validation



Prototyping activities

- Choose prototype testers
  - The best testers are users who are fairly experienced and who are open-minded about the use of new systems. End-users who do different jobs should be involved so that different areas of system functionality will be covered.
- Develop test scenarios
  - Careful planning is required to draw up a set of test scenarios which provide broad coverage of the requirements. End-users shouldn’t just play around with the system as this may never exercise critical system features.
- Execute scenarios
  - The users of the system work, usually on their own, to try the system by executing the planned scenarios.
- Document problems
  - Its usually best to define some kind of electronic or paper problem report form which users fill in when they encounter a problem.

User manual development

- Writing a user manual from the requirements forces a detailed requirements analysis and thus can reveal problems with the document
- Information in the user manual
  - Description of the functionality and how it is implemented
  - Which parts of the system have not been implemented
  - How to get out of trouble
  - How to install and get started with the system

Model validation

- Validation of system models is an essential part of the validation process
- Objectives of model validation
- To demonstrate that each model is self-consistent
- If there are several models of the system, to demonstrate that these are internally and externally consistent
- To demonstrate that the models accurately reflect the real requirements of system stakeholders
- Some checking is possible with automated tools
- Paraphrasing the model is an effective checking technique

Requirements testing

- Each requirement should be testable i.e. it should be possible to define tests to check whether or not that requirement has been met.
- Inventing requirements tests is an effective validation technique as missing or ambiguous information in the requirements description may make it difficult to formulate tests
- Each functional requirement should have an associated test

Test case definition

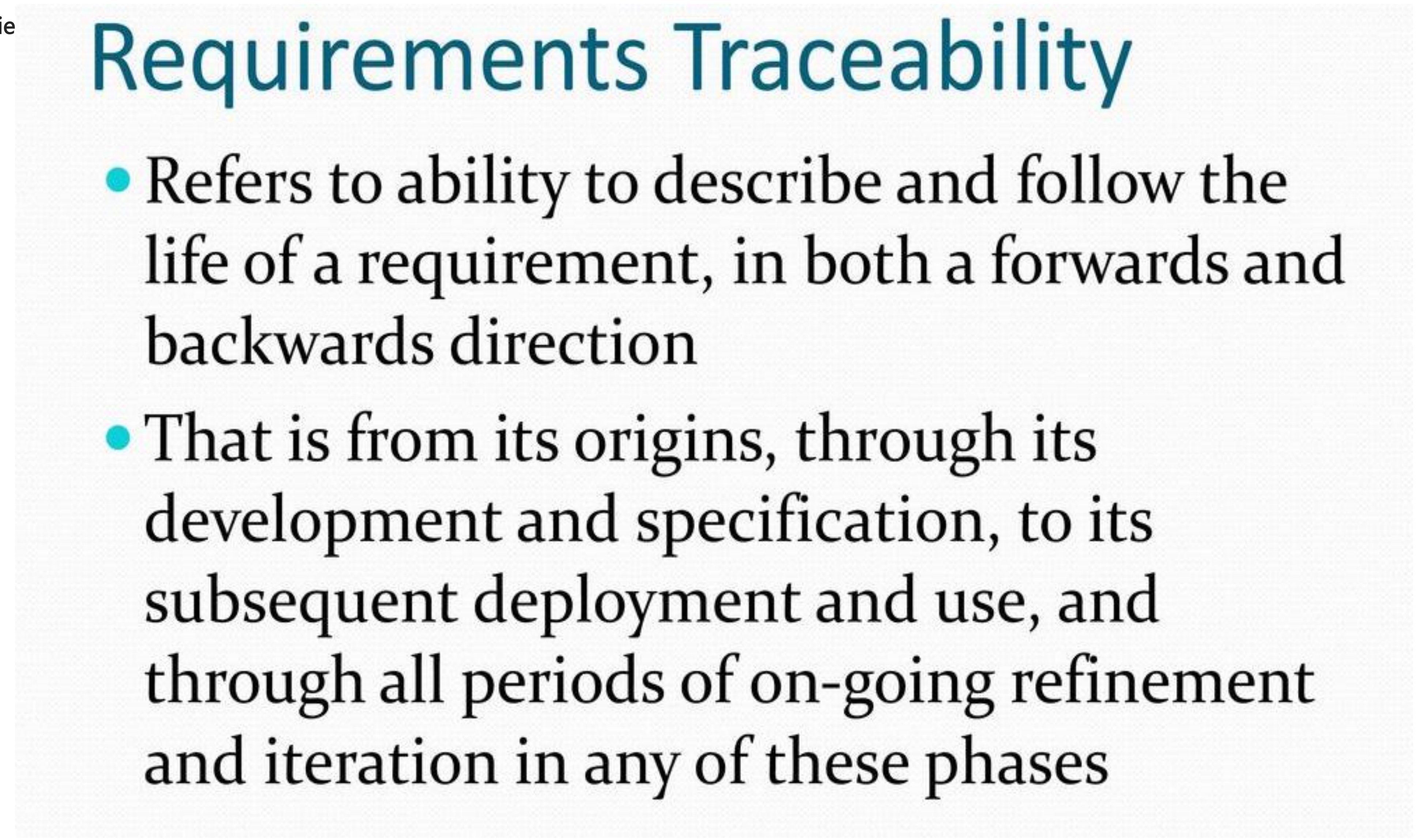
- What usage scenario might be used to check the requirement?
- Does the requirement, on its own, include enough information to allow a test to be defined?
- Is it possible to test the requirement using a single test or are multiple test cases required?
- Could the requirement be re-stated to make the test cases more obvious?

Test record form

- The requirement's identifier
  - There should be at least one for each requirement.
- Related requirements
  - These should be referenced as the test may also be relevant to these requirements.
- Test description
  - A brief description of the test and why this is an objective requirements test. This should include system inputs and corresponding outputs.
- Requirements problems
  - A description of problems which made test definition difficult or impossible.
- Comments and recommendations
  - These are advice on how to solve requirements problems which have been discovered.

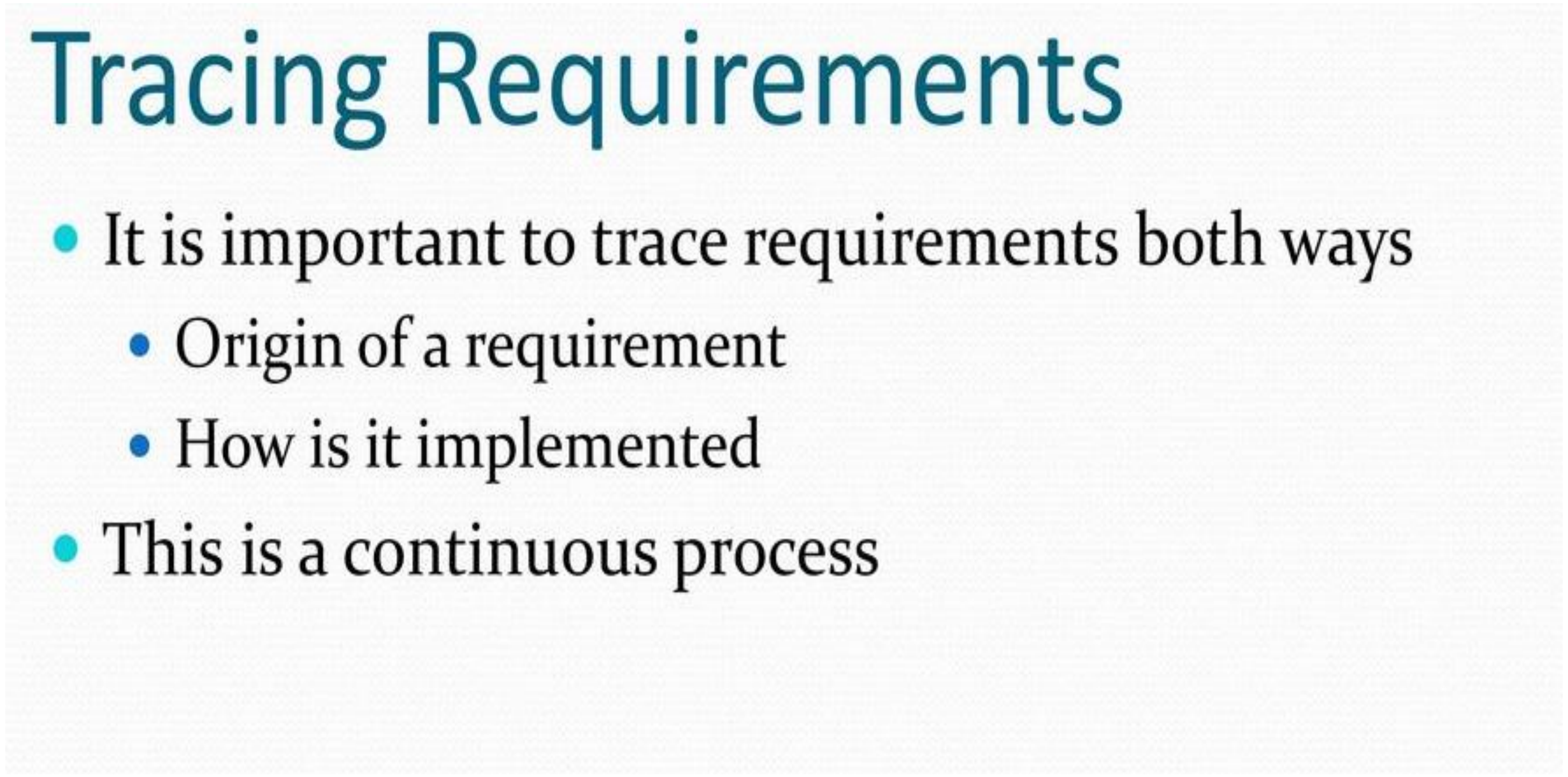
REQUIREMENTS TRACEABILITY

- Property of an element.f document or code that indictaes the degree to which it can be traced to its origin or "reason for being".
- Ability to establish a predecessor-successor relationship between one work product to another.
- A product can said to be traceable if it can be proved that it complies with its specifications
- satisfies all requirement of customer for design.
- consern with relationship of requirements their sources & the system design.
- Using this requirements finding becomes easy.



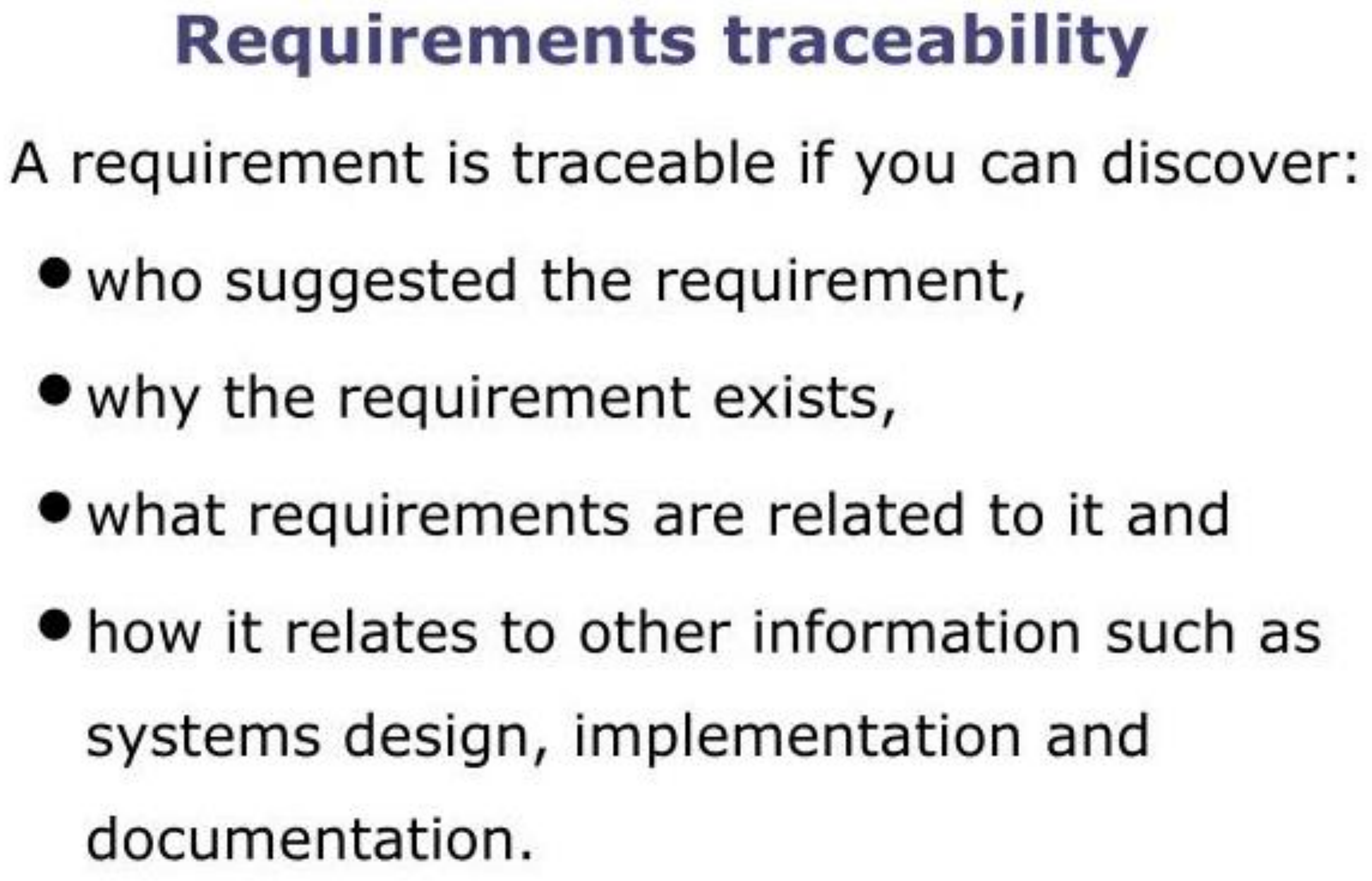
Requirements Traceability

- Refers to ability to describe and follow the life of a requirement, in both a forwards and backwards direction
- That is from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases



Tracing Requirements

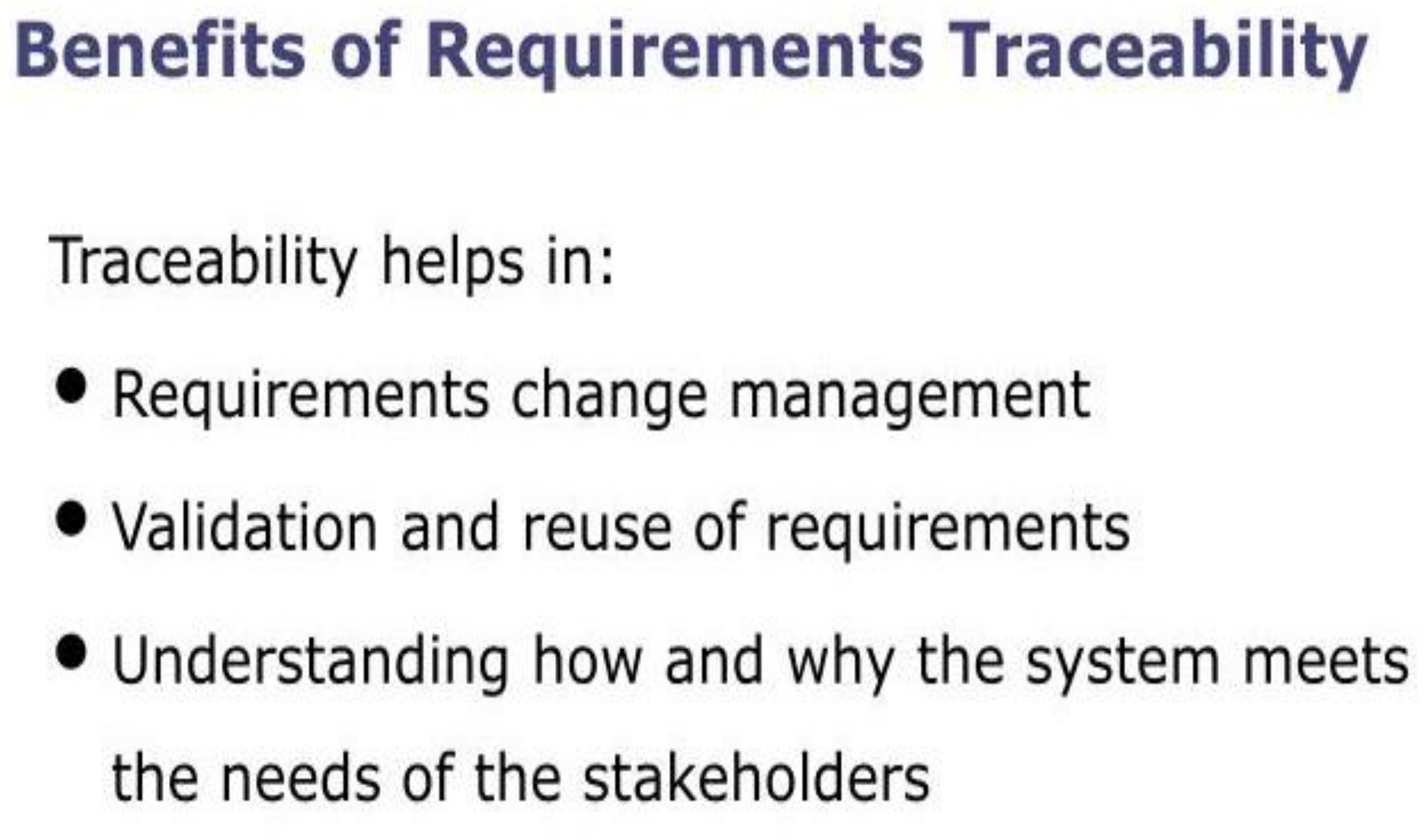
- It is important to trace requirements both ways
  - Origin of a requirement
  - How is it implemented
- This is a continuous process



Requirements traceability

A requirement is traceable if you can discover:

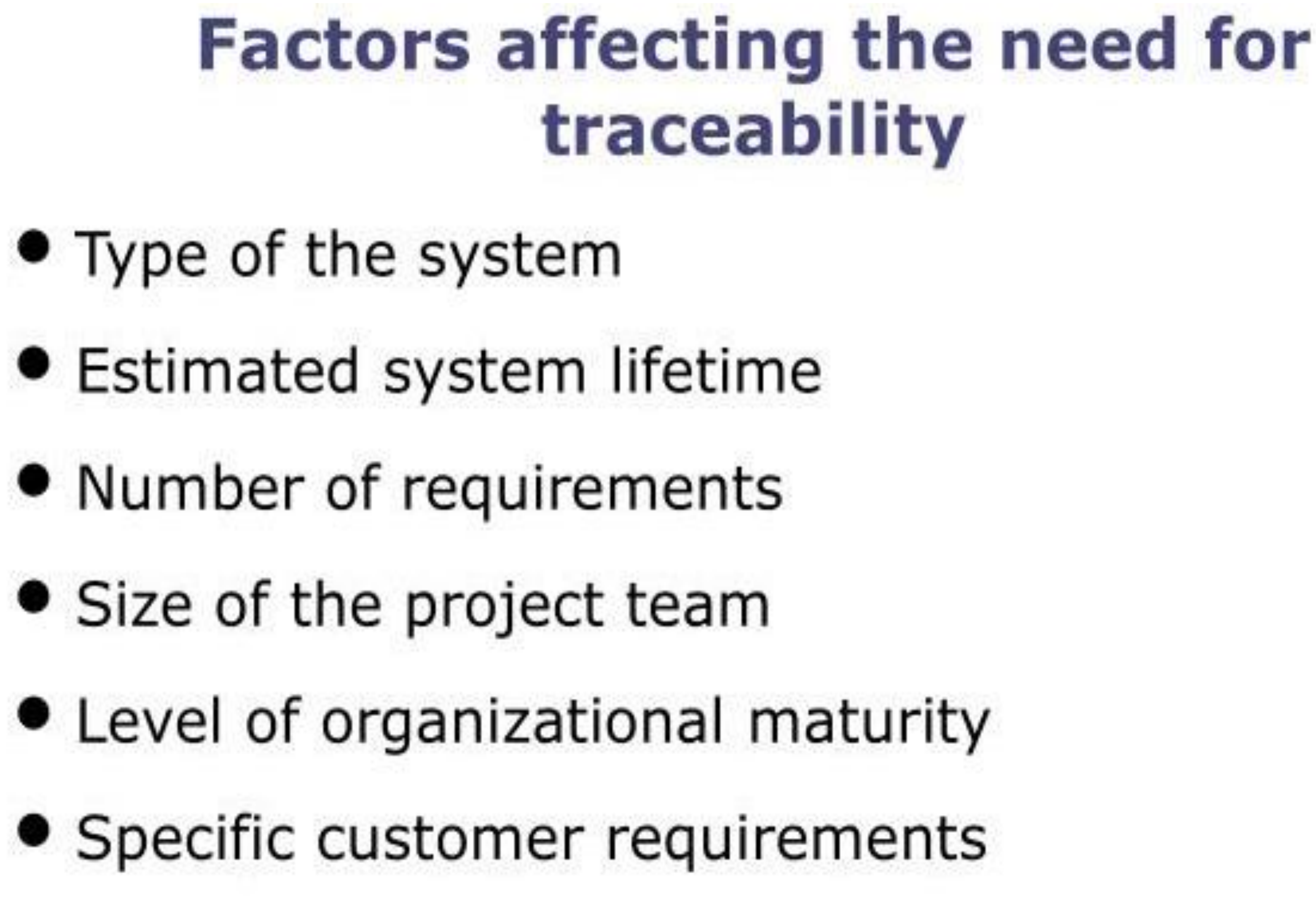
- who suggested the requirement,
- why the requirement exists,
- what requirements are related to it and
- how it relates to other information such as systems design, implementation and documentation.



Benefits of Requirements Traceability

Traceability helps in:

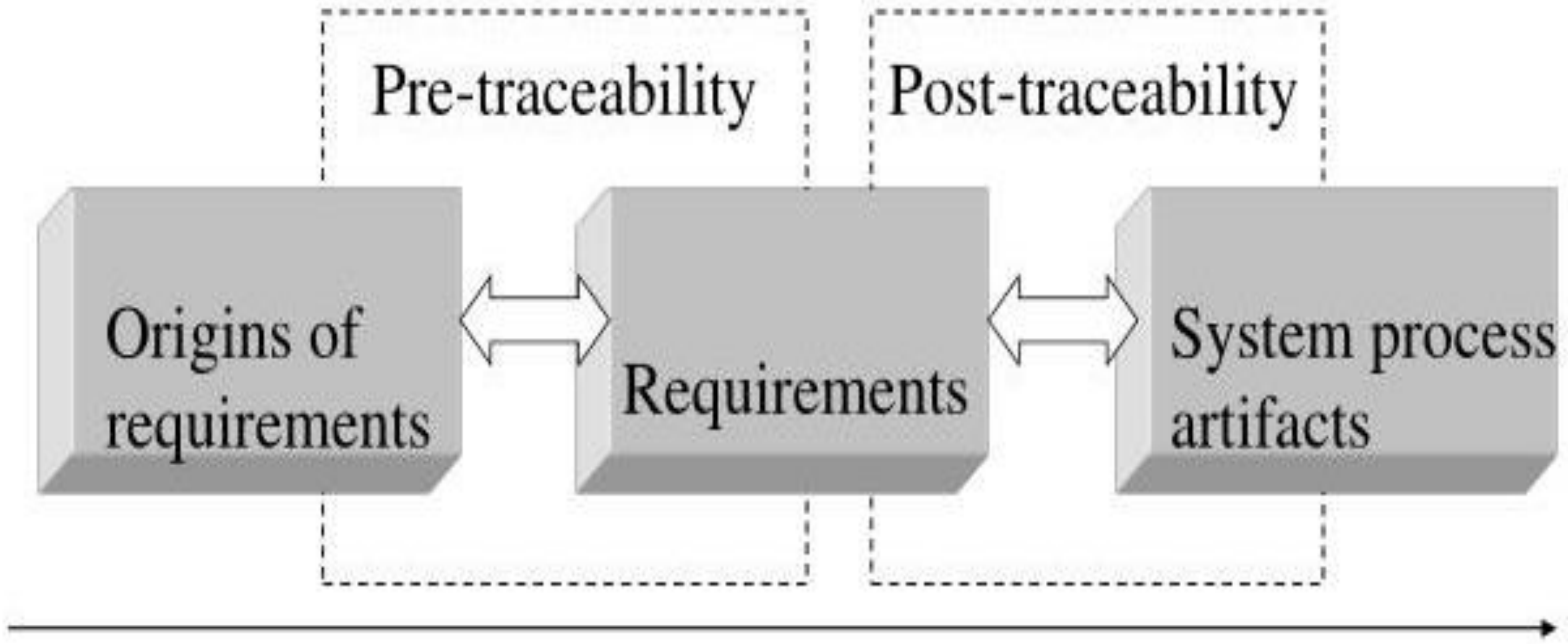
- Requirements change management
- Validation and reuse of requirements
- Understanding how and why the system meets the needs of the stakeholders



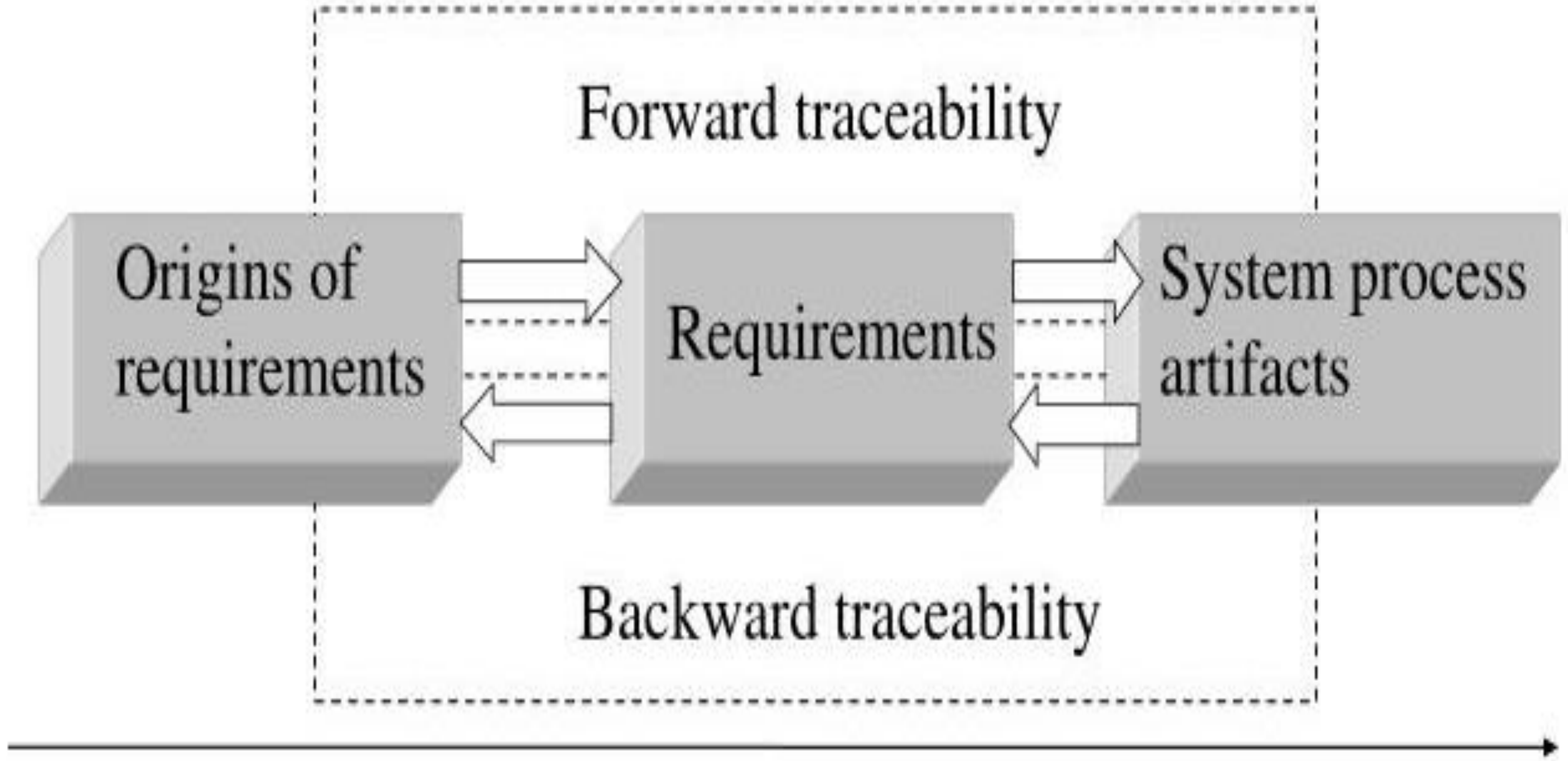
Factors affecting the need for traceability

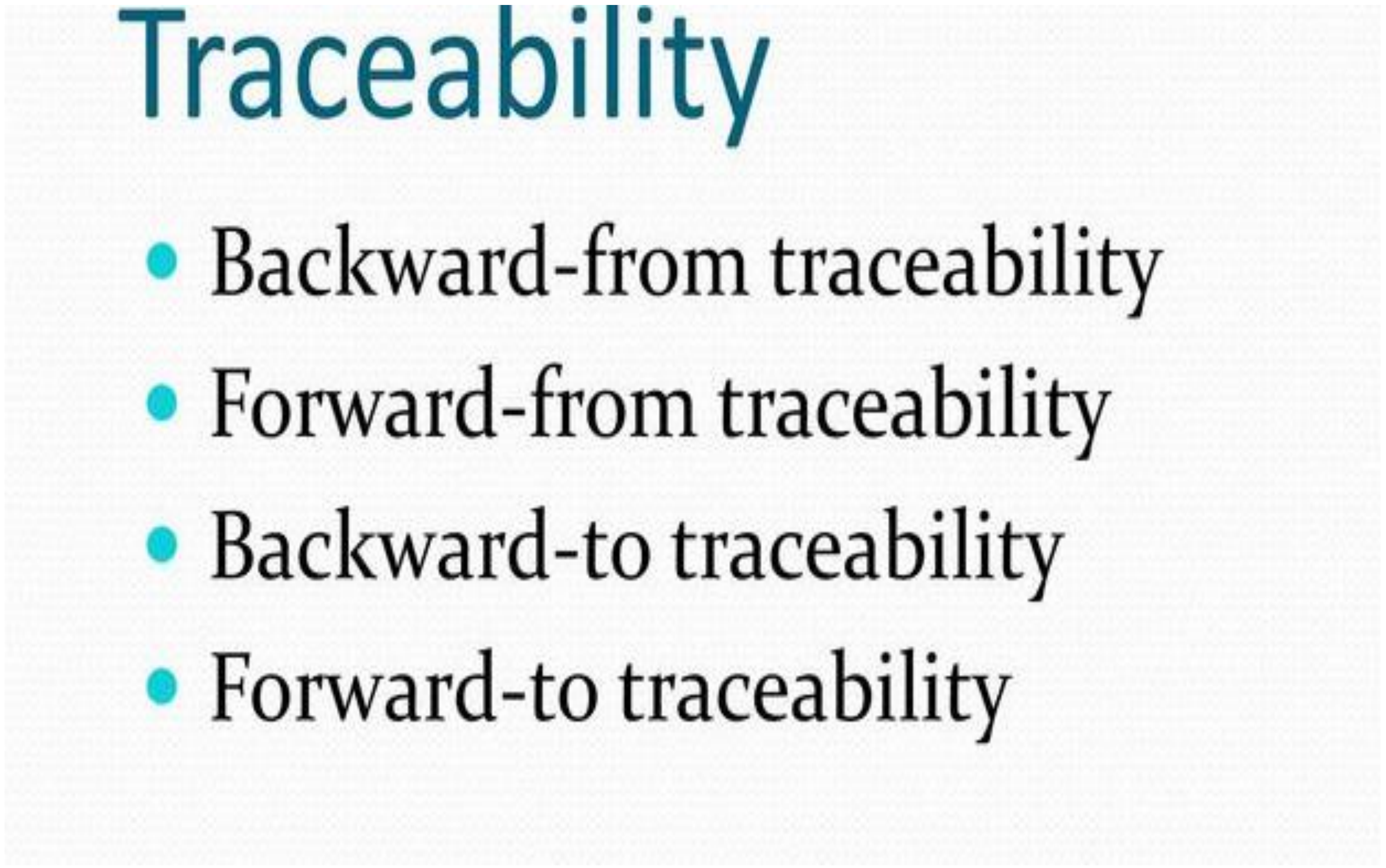
- Type of the system
- Estimated system lifetime
- Number of requirements
- Size of the project team
- Level of organizational maturity
- Specific customer requirements

Traceability classes 1/2
Pre- and post-traceability



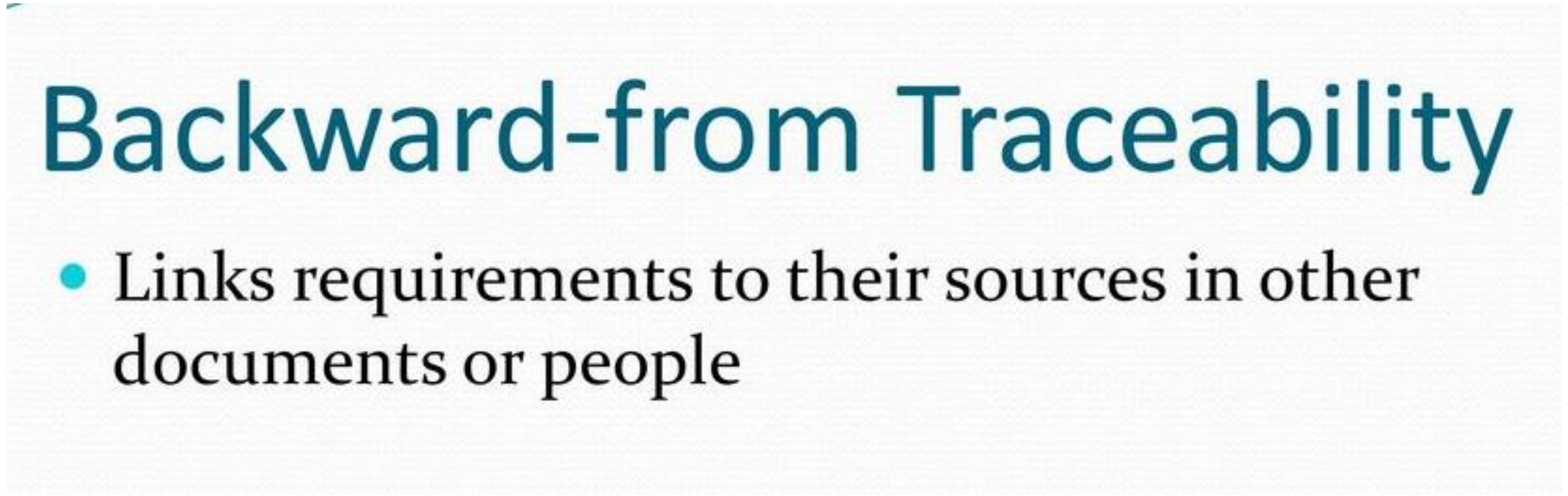
Traceability classes 2/2
Forward and backward traceability





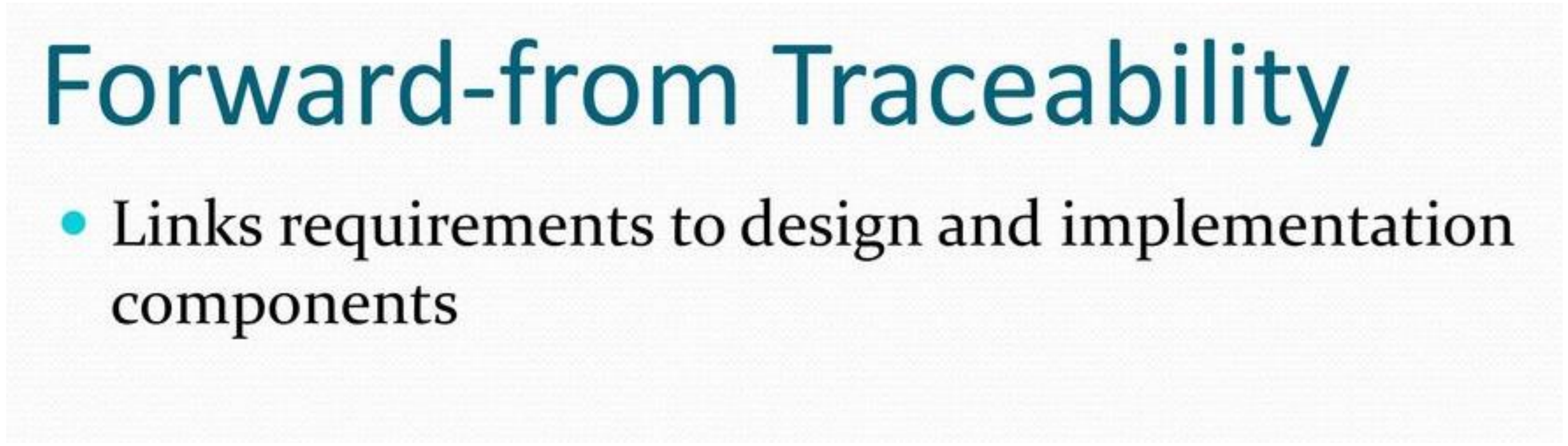
Traceability

- Backward-from traceability
- Forward-from traceability
- Backward-to traceability
- Forward-to traceability



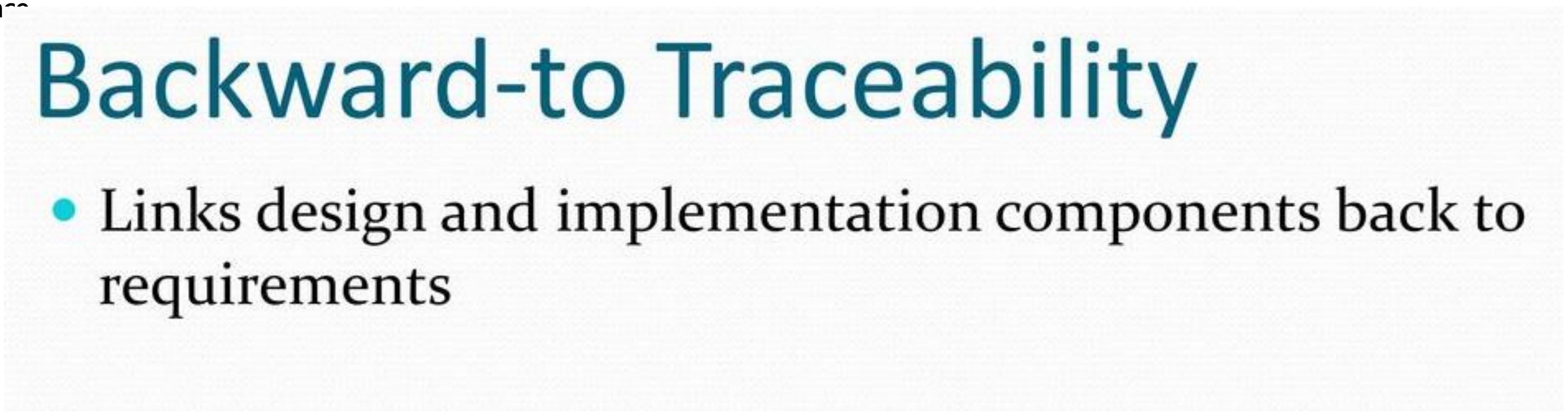
Backward-from Traceability

- Links requirements to their sources in other documents or people



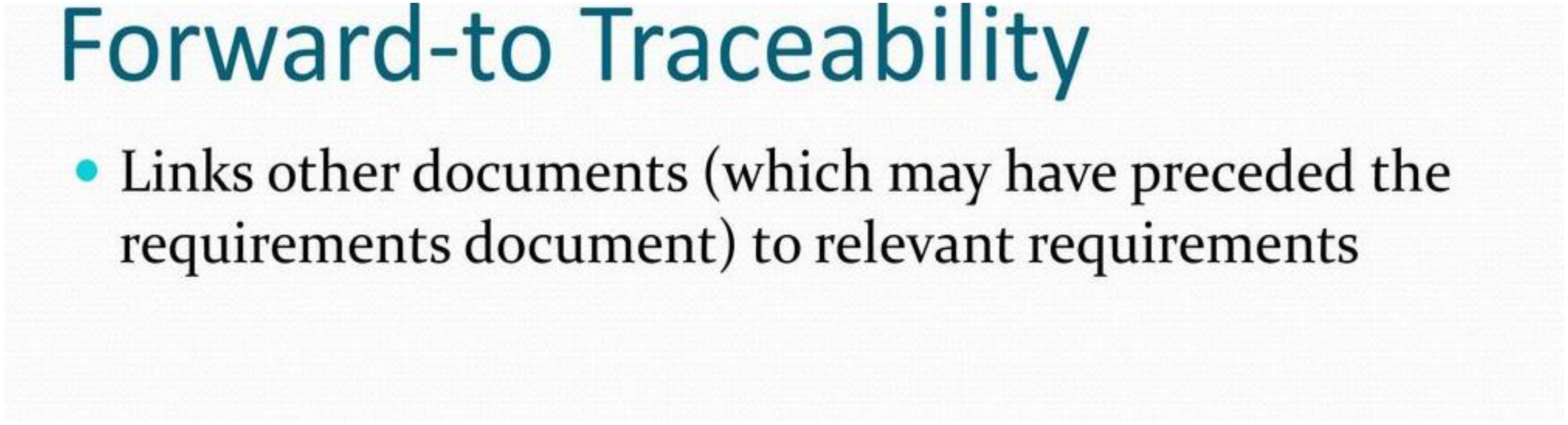
Forward-from Traceability

- Links requirements to design and implementation components



Backward-to Traceability

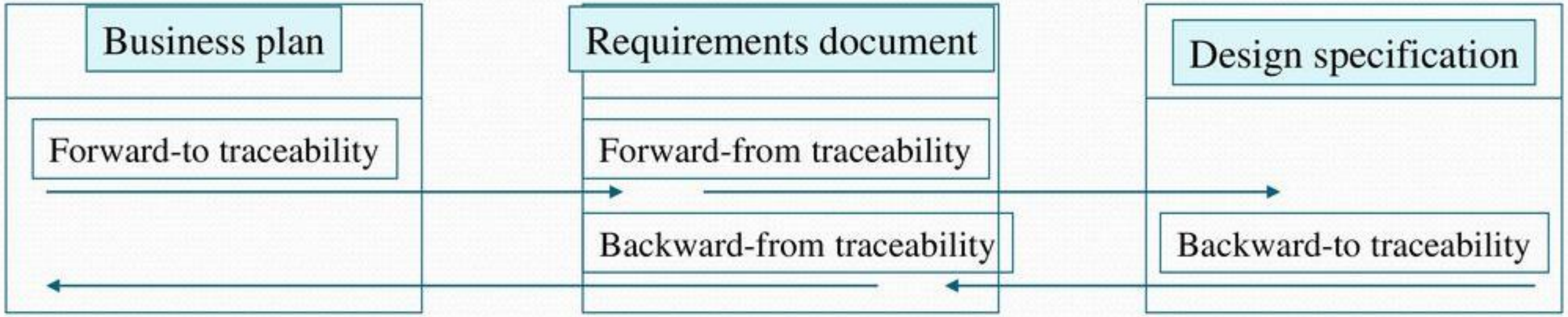
- Links design and implementation components back to requirements



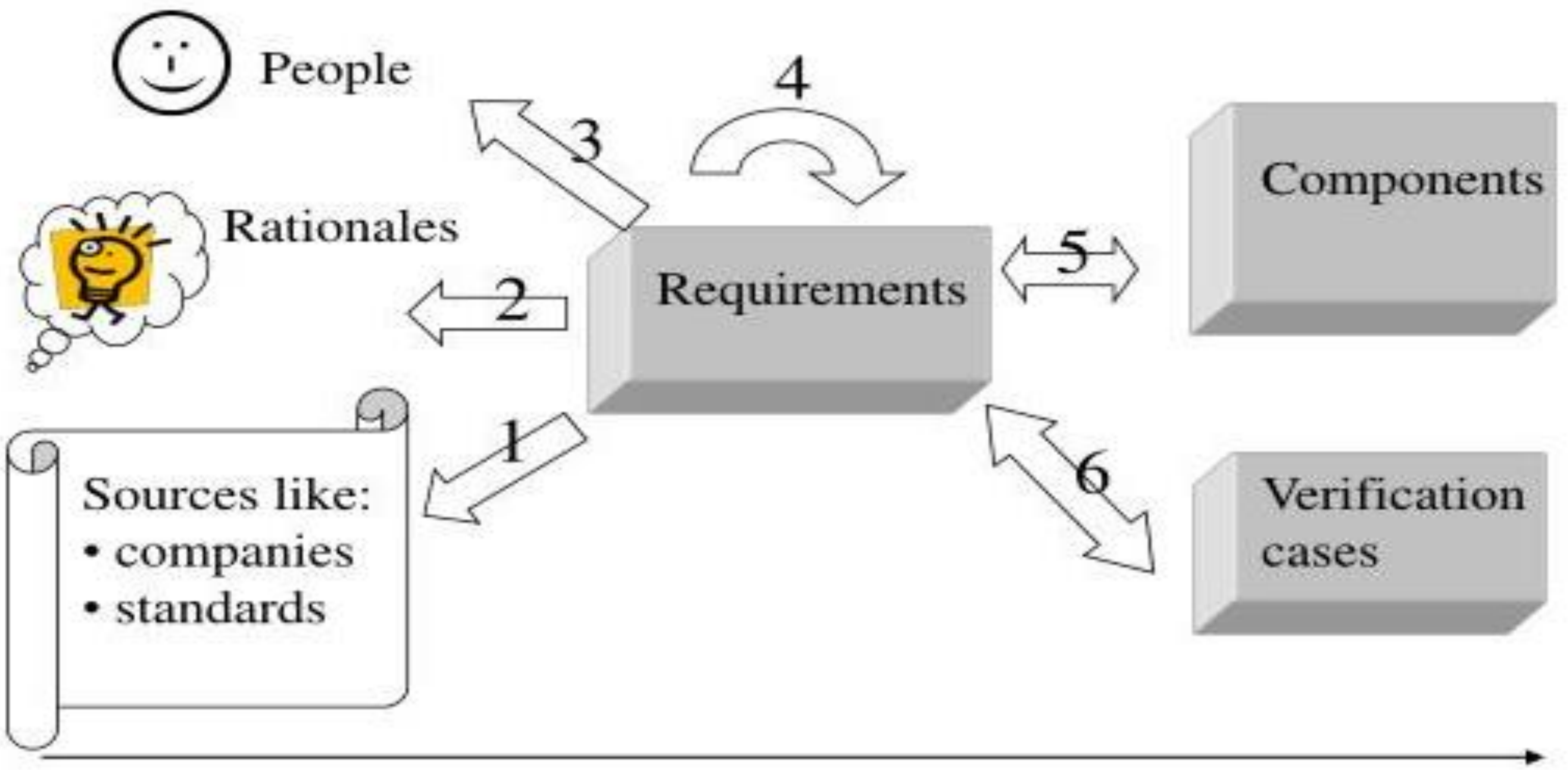
Forward-to Traceability

- Links other documents (which may have preceded the requirements document) to relevant requirements

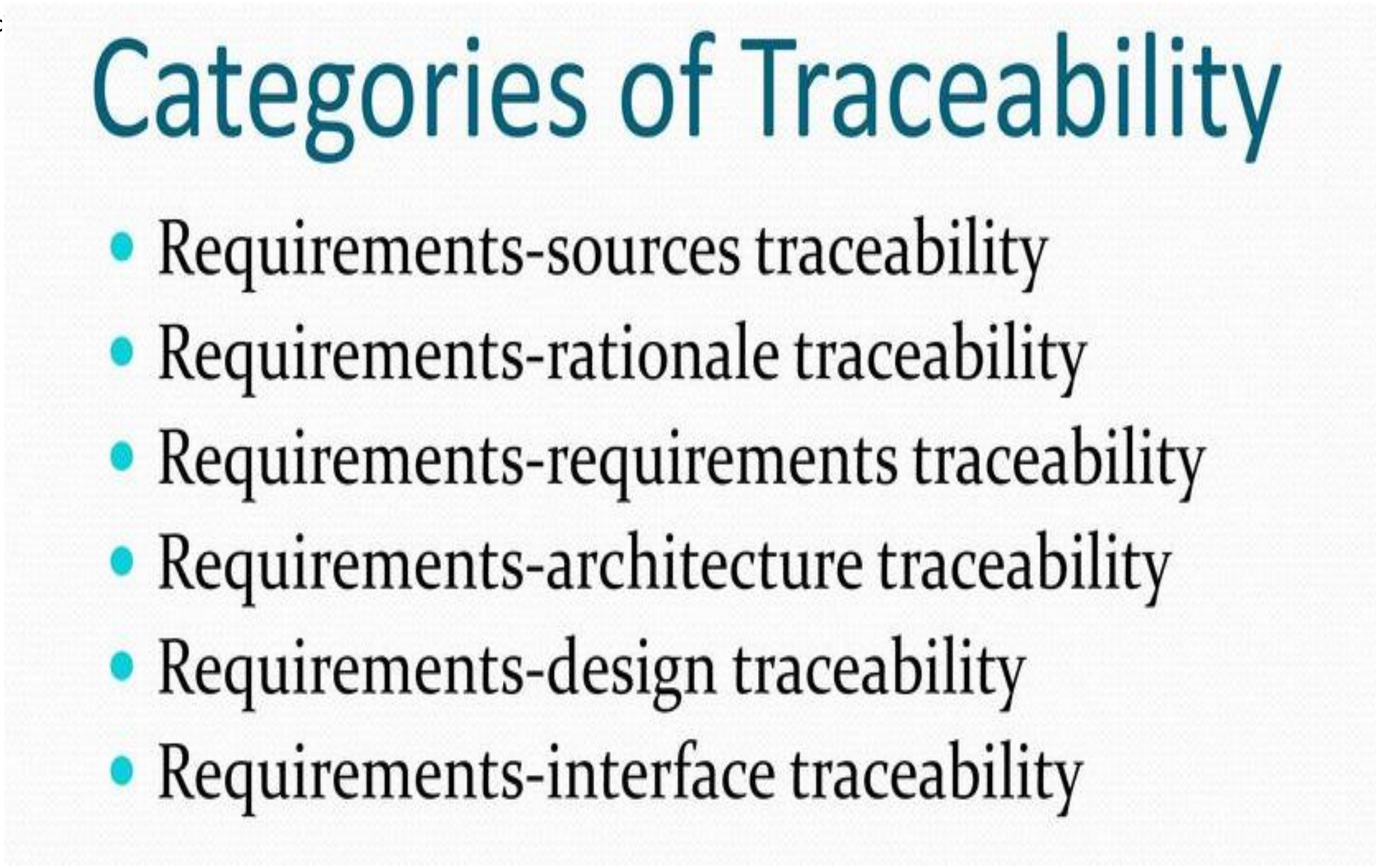
Backwards and Forwards Traceability



Traceability relation types

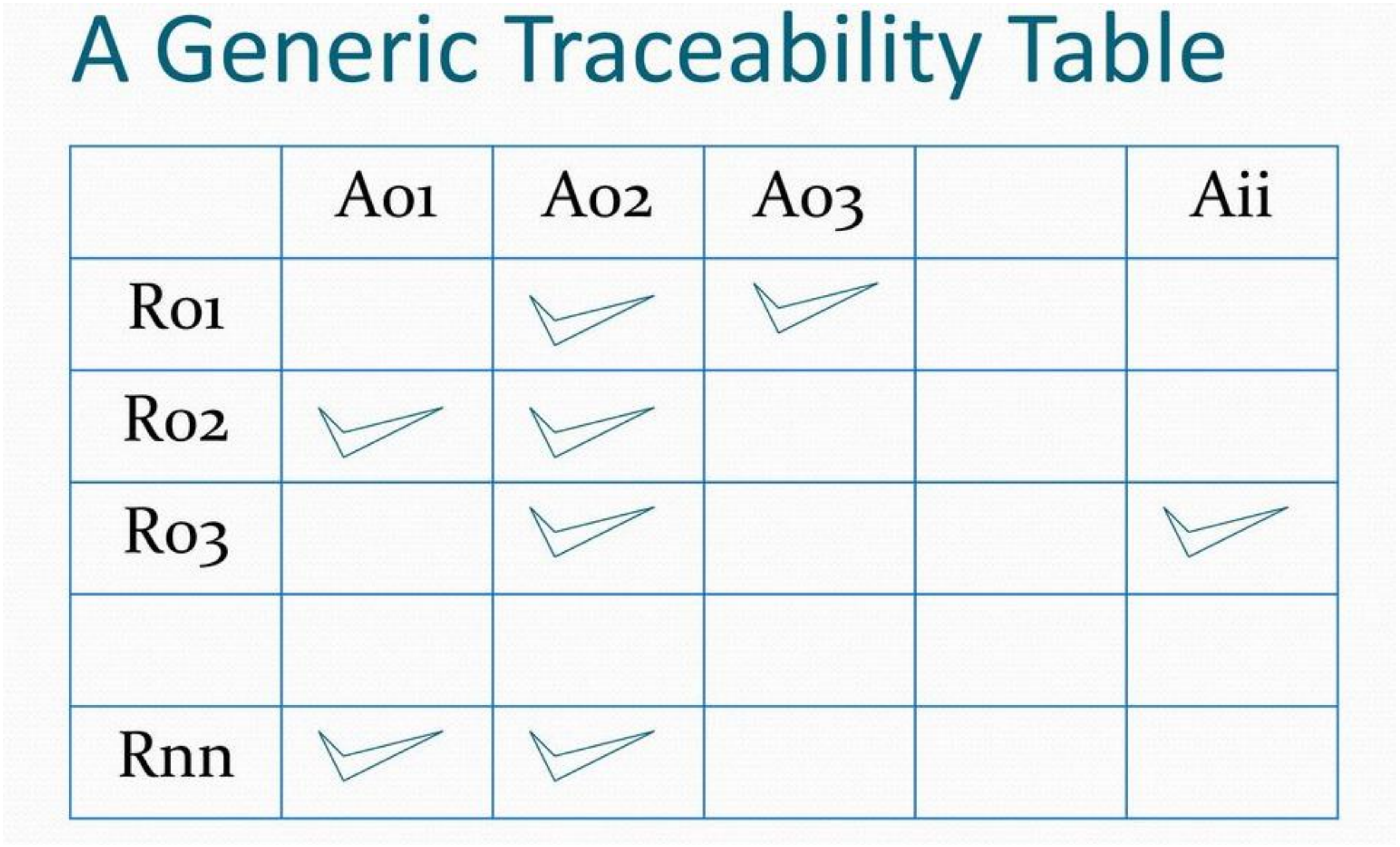


1) requirement-source
2) requirement-rationale
3) requirement-people
4) requirement-requirement
5) requirement-component
6) requirement-verification



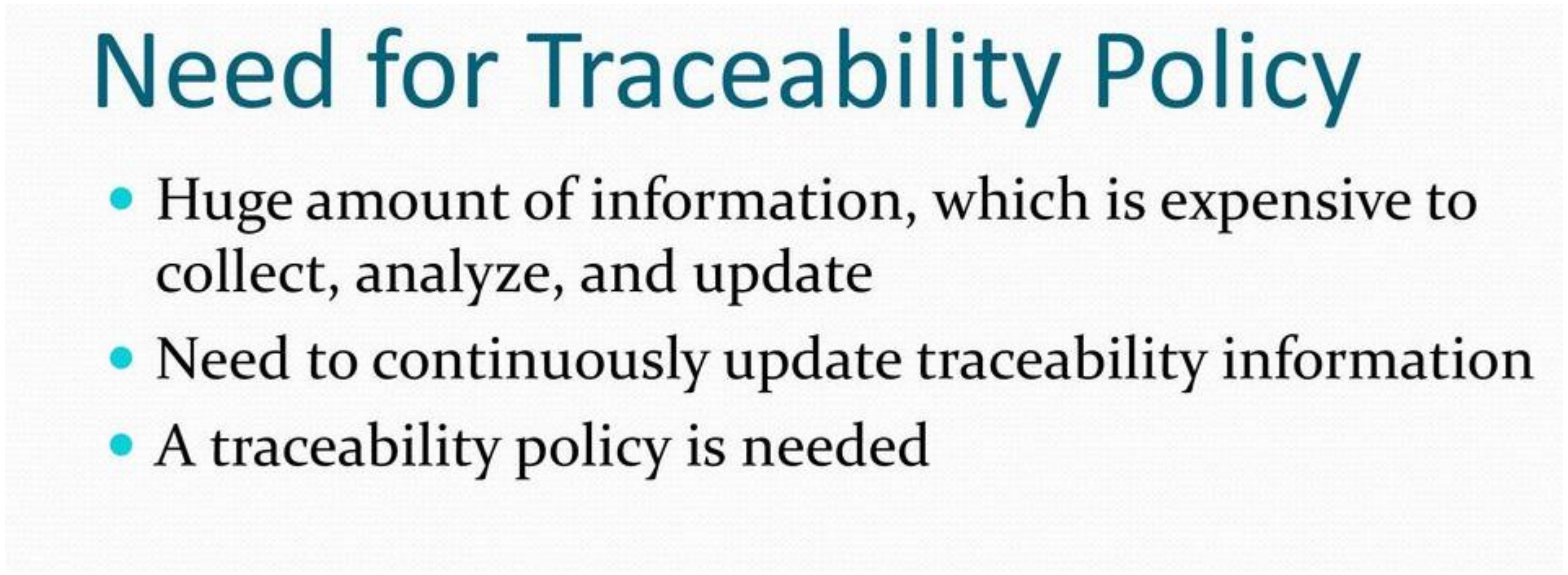
Categories of Traceability

- Requirements-sources traceability
- Requirements-rationale traceability
- Requirements-requirements traceability
- Requirements-architecture traceability
- Requirements-design traceability
- Requirements-interface traceability



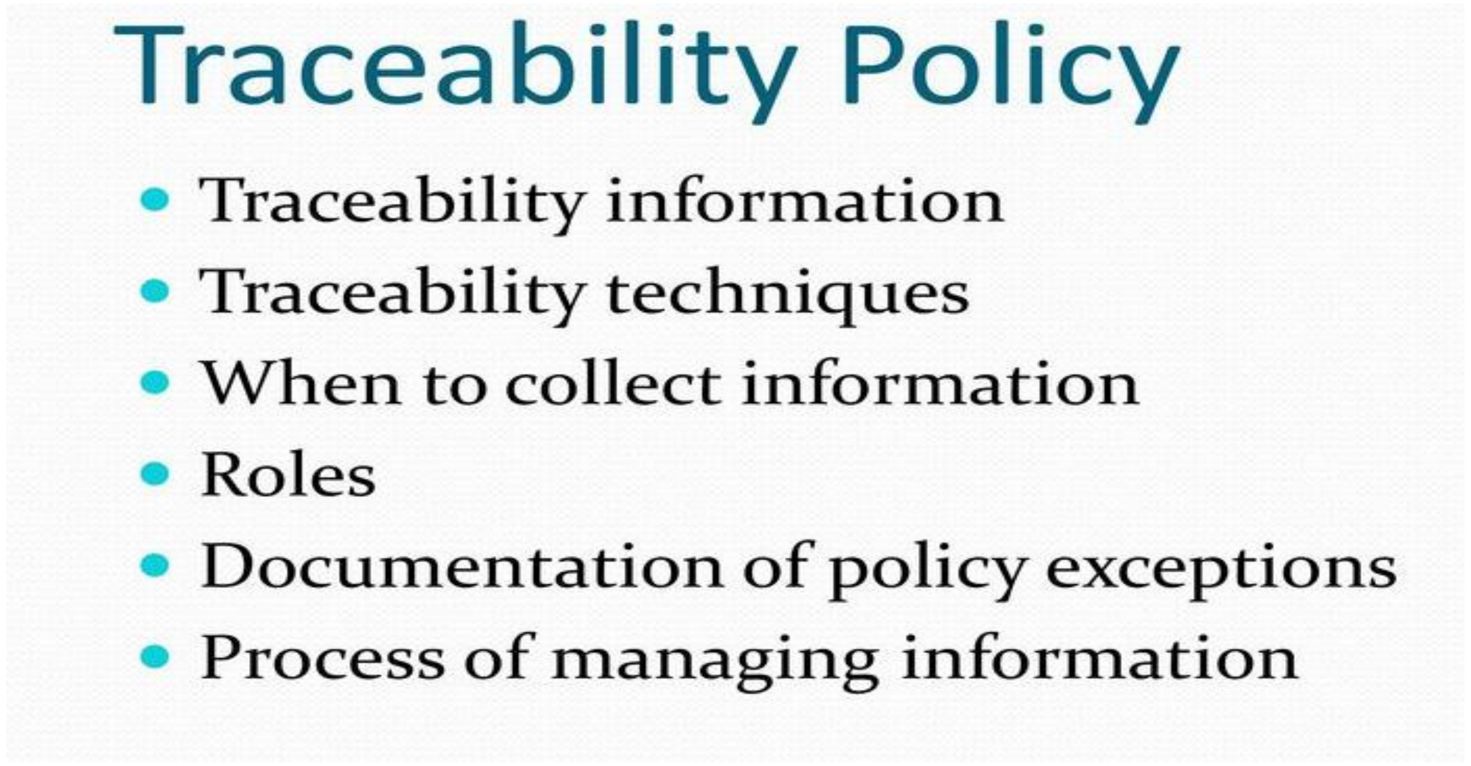
A Generic Traceability Table

	A01	A02	A03		Aii
R01		✓	✓		
R02	✓	✓			
R03		✓			✓
Rnn	✓	✓			



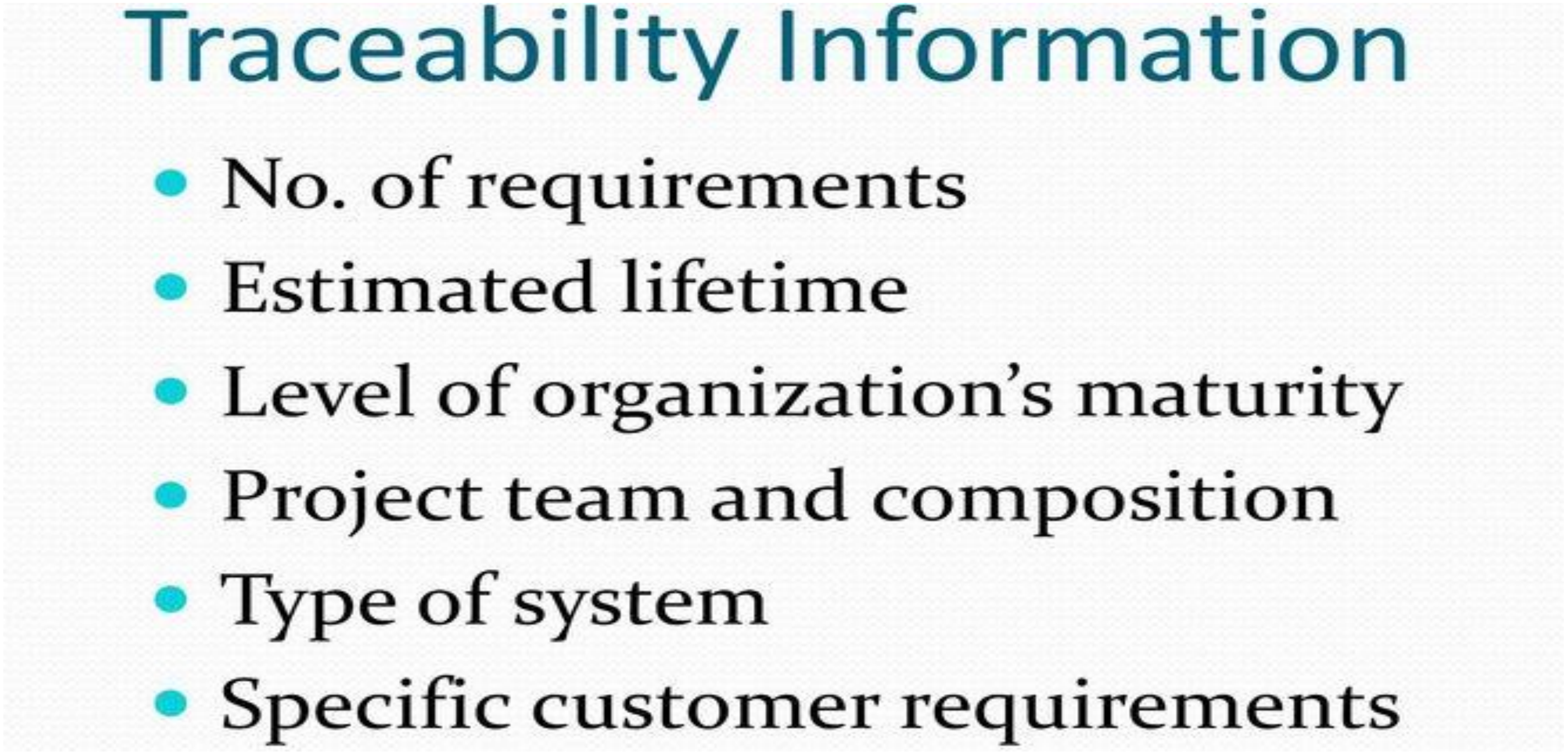
Need for Traceability Policy

- Huge amount of information, which is expensive to collect, analyze, and update
- Need to continuously update traceability information
- A traceability policy is needed



Traceability Policy

- Traceability information
- Traceability techniques
- When to collect information
- Roles
- Documentation of policy exceptions
- Process of managing information



Traceability Information

- No. of requirements
- Estimated lifetime
- Level of organization’s maturity
- Project team and composition
- Type of system
- Specific customer requirements

ANY QUERY