UNIT II Software Requirements and Requirements Engineering Process

Software Requirements

Software Requirements

> Software Requirements
>
> # Software Requirements
>
> Software requirements are essential to:
>
> - Introduce the concepts of user and system requirements
> - Describe functional and non-functional requirements
> - Explain how software requirements may be organized in a requirements document
>
> ## What is a Requirement?
>
> A requirement for the system entails the description of the services provided by the system and its operational constraints. It can range from a high-level abstract statement of a service or system constraint to a detailed mathematical functional specification. Requirements may serve a dual function:
>
> - The basis for a bid for a contract, thus must be open to interpretation.
> - The basis for the contract itself, therefore must be defined in detail.
>
> Both these statements may be called requirements.
>
> ## Requirements Engineering
>
> Requirements engineering is the process of finding out, analyzing, documenting, and checking services and constraints. It involves establishing the services that the customer requires from a system and the constraints under which it operates and is developed. The requirements themselves are the descriptions of the system services and constraints generated during the requirements engineering process.
>
> ### Requirements Abstraction (Davis)
>
> When letting a contract for a large software development project, a company must define its needs in an abstract way to allow multiple contractors to bid, offering

different solutions. After a contract is awarded, the contractor writes a more detailed system definition for the client to understand and validate, and both documents may be called the requirements document for the system.

Types of Requirement

- 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 outlining detailed descriptions of the system's functions, services, and operational constraints. It defines what should be implemented and may be part of a contract between the client and contractor.

Definitions and Specifications

- User Requirement Definition: The software must provide the means of representing and accessing external files created by other tools.
- System Requirement Specification:
    - The user should be provided with facilities to define the type of external files.
    - Each external file type may have an associated tool that may be applied to the file.
    - Each external file type may be represented as a specific icon on the user's display.
    - Facilities should be provided for the icon representing an external file type to be defined by the user.
    - When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected icon.

Requirements Readers

Functional and Non-functional Requirements

- Functional Requirements: Statements of services the system should provide, specifying how the system should react to particular inputs and behave in specific situations.
- Non-functional Requirements: Constraints on the services or functions offered by the system, including timing constraints, development process constraints, standards, etc.

- Domain Requirements: Requirements from the application domain of the system reflecting characteristics of that domain.

Functional and Non-functional Requirements

Functional and Non-functional Requirements

Functional Requirements

Functional requirements describe the functionality or system services, depending on the type of software, expected users, and the system's type of use. While functional user requirements may be high-level statements of what the system should do, functional system requirements should provide a detailed description of the system services.

LIBSYS System Functional Requirements

The LIBSYS system, a library system providing a single interface to databases of articles in different libraries, has the following functional requirements:

1. The user shall be able to search either all of the initial set of databases or select a subset from it.
2. The system shall provide appropriate viewers for the user to read documents in the document store.
3. Every order shall be allocated a unique identifier (ORDER_ID), which the user shall be able to copy to the account's permanent storage area.

Requirements Imprecision

Problems arise when requirements are not precisely stated. Ambiguous requirements may be interpreted differently by developers and users. For example, the term 'appropriate viewers' can be interpreted differently:

- User intention: Special purpose viewer for each different document type.
- Developer interpretation: Provide a text viewer that shows the contents of the document.

Requirements Completeness and Consistency

In principle, requirements should be both complete and consistent:

- Complete: They should include descriptions of all required facilities.
- Consistent: There should be no conflicts or contradictions in the descriptions of the system facilities. However, in practice, producing a complete and

consistent requirements document is challenging.

Non-functional Requirements

Non-functional requirements define system properties and constraints such as reliability, response time, and storage requirements. These requirements may be more critical than functional requirements, as the system becomes useless if they are not met.

Non-functional Requirement Types

1. Product Requirements: Specifications that the delivered product must behave in a particular way (e.g., execution speed, reliability). Example: The user interface for LIBSYS shall be implemented as simple HTML without frames or Java applets.
2. Organisational Requirements: Consequences of organizational policies and procedures (e.g., process standards, implementation requirements). Example: The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95.
3. External Requirements: Arise from factors external to the system and its development process (e.g., interoperability requirements, legislative requirements). Example: The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.

Goals and Requirements

Non-functional requirements, especially goals, may be challenging to state precisely and verify. Goals express general user intentions, while verifiable non-functional requirements provide measurable statements for testing.

- Goal: The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized.
- Verifiable Non-functional Requirement: Experienced controllers shall be able to use all system functions after a total of two hours of training. After training, the average number of errors made by experienced users shall not exceed two per day.

Requirements Measures

Property	Measure
Speed	Processed transactions/second, User/Event response time, Screen

Property	Measure
	refresh time
Size	M Bytes, Number of ROM chips
Ease of use	Training time, Number of help frames
Reliability	Mean time to failure, Probability of unavailability, Rate of failure occurrence, Availability
Robustness	Time to restart after failure, Percentage of events causing failure, Probability of data corruption on failure
Portability	Percentage of target-dependent statements, Number of target systems

Requirements Interaction

Conflicts between different non-functional requirements are common in complex systems, as illustrated by the example of a spacecraft system trying to minimize weight and power consumption simultaneously. Resolving such conflicts is crucial for successful system development.

A common challenge with non-functional requirements is their difficulty to verify, often stated as vague goals. Users may express these requirements as general goals, leaving room for interpretation and potential disputes after system delivery.

User Requirements

User Requirements

User requirements serve as a bridge between system developers and end-users who may not possess detailed technical knowledge. It is essential to describe both functional and non-functional requirements in a way that is easily understandable by users. This involves utilizing natural language, tables, and diagrams to ensure clarity for all users.

Challenges with Natural Language in User Requirements

1. Lack of Clarity: Achieving precision without making the document difficult to read can be challenging.
2. Requirements Confusion: Functional and non-functional requirements may become intertwined.
3. Requirements Amalgamation: Different requirements may be expressed together, leading to confusion.

Problems in User Requirement Descriptions

1. Database Requirements: Combining conceptual and detailed information, such as describing the concept of a financial accounting system in LIBSYS. While the concept is necessary, including unnecessary details like manager configuration at this level is confusing.
2. Grid Requirement: Mixing different kinds of requirements, including conceptual functional requirements (the need for a grid), non-functional requirements (grid units), and non-functional UI requirements (grid switching).

Structured Presentation of User Requirements

Guidelines for Writing Requirements

1. Invent a Standard Format: Establish a standard format and consistently apply it to all requirements.
2. Use Consistent Language: Use "shall" for mandatory requirements and "should" for desirable requirements.
3. Text Highlighting: Utilize text highlighting to identify key parts of the requirement.
4. Avoid Computer Jargon: Ensure the avoidance of computer jargon for better user comprehension.

By following these guidelines, user requirements can be presented in a structured and understandable manner, fostering effective communication between system developers and end-users.

System Requirements

System Requirements

System Requirements

System requirements provide more detailed specifications of system functions, services, and constraints than user requirements. They serve as the basis for designing the system and may be incorporated into the system contract. These requirements can be defined or illustrated using system models.

Relationship Between Requirements and Design

In principle, requirements state what the system should do, while design describes how it achieves this. However, in practice, requirements and design are inseparable. A system architecture may be designed to structure the requirements, and specific designs may be influenced by domain requirements.

Challenges with Natural Language (NL) Specification

1. Ambiguity: NL is naturally ambiguous, making it challenging for both readers and writers to interpret the same words in the same way.
2. Over-flexibility: The same concept may be expressed in various ways in the specification.
3. Lack of Modularization: NL structures are often inadequate for structuring system requirements.

Alternatives to NL Specification

1. Structured Natural Language Specifications: This approach relies on defining standard forms or templates to express requirements.
2. Design Description Languages: Using a language resembling a programming language with more abstract features to define an operational model of the system.
3. Graphical Notations: Utilizing graphical languages, supplemented by text annotations, to define functional requirements. Examples include use-case descriptions and sequence diagrams.
4. Mathematical Specifications: Notations based on mathematical concepts, such as finite-state machines or sets, providing unambiguous specifications. However, customers may find formal specifications challenging to understand.

3.1) Structured Language Specifications

- Freedom Limitation: Writers are limited by predefined templates.
- Standardization: Requirements are written uniformly.

Form-based Specifications

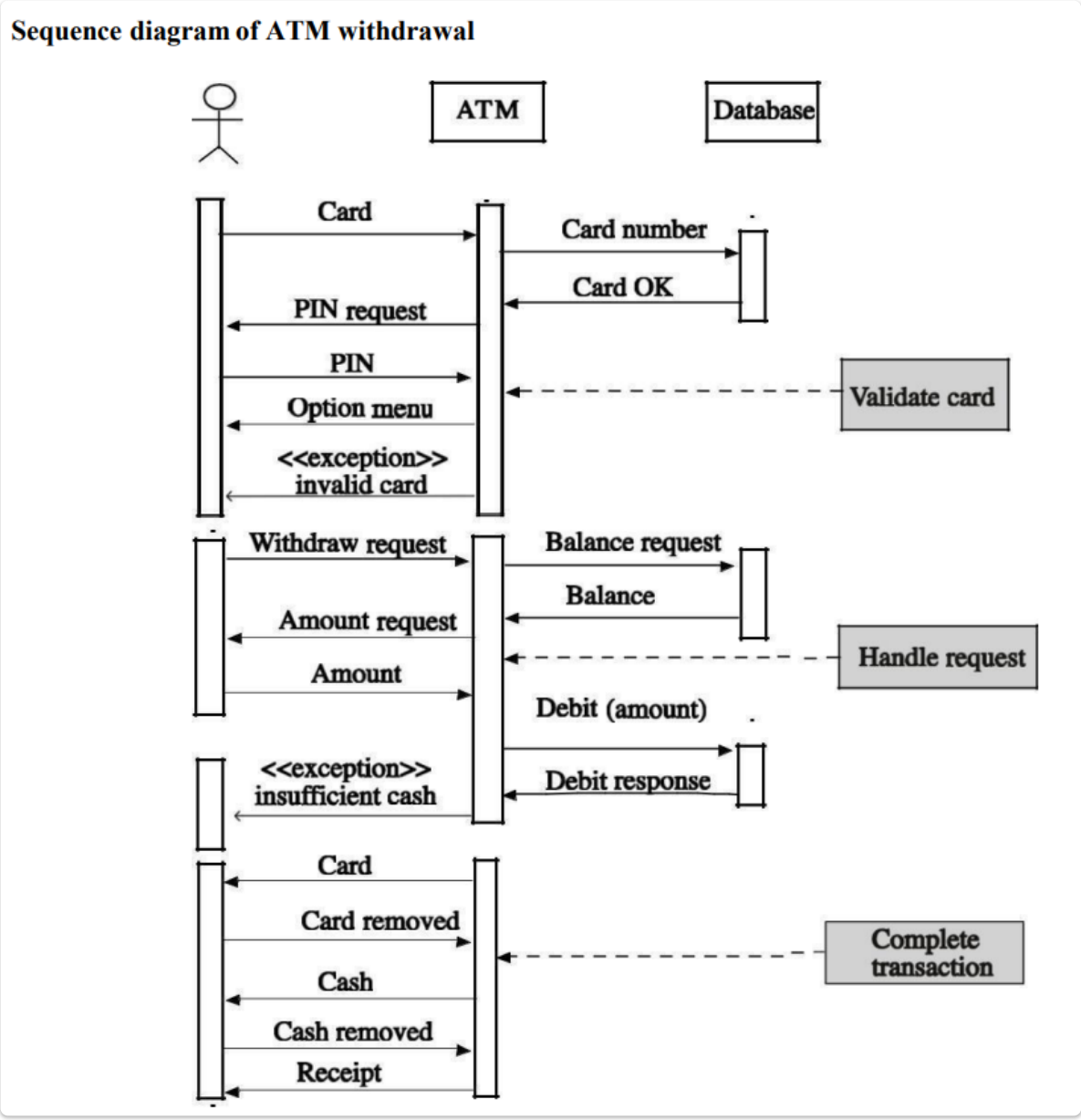
- Definition of the function or entity.
- Description of inputs and their sources.
- Description of outputs and their destinations.
- Indication of other required entities.
- Pre and post-conditions.
- Side effects.

Tabular Specifications

- Useful for defining alternative courses of action.

Graphical Models

- Useful for showing state changes or describing a sequence of actions.
- Examples include sequence diagrams.



System Requirement Specification Using a Standard Form

- Function: Description of the function or entity.
- Inputs: Description of inputs and their sources.
- Outputs: Description of outputs and their destinations.
- Action: Description of the action to be taken.
- Requires: Indication of other entities used.
- Pre-condition: Setting out what must be true before the function is called.
- Post-condition: Specifying what is true after the function is called.
- Side-effects: Description of the side effects of the operation.

Interface Specification

Interface Specification

In software engineering, the interface specification is crucial, as most systems need to operate seamlessly with other systems. The interfaces through which systems interact must be explicitly defined as part of the requirements. There are three main types of interfaces that may need to be specified:

1. Procedural Interfaces

These interfaces involve existing programs or subsystems that provide a set of services accessible through interface procedures. Such interfaces are commonly referred to as **Application Programming Interfaces (APIs)**.

2. Data Structures

Interfaces may involve the exchange of data structures between subsystems. Graphical data models are often the most effective notations for describing this type of interface.

3. Data Representations

This type of interface specification deals with established data representations for existing subsystems. Formal notations are highly effective in specifying these interfaces.

In summary, a well-defined interface specification is essential for ensuring smooth interactions between different software components or systems. Formal notations and graphical data models play a crucial role in accurately representing procedural interfaces, data structures, and data representations in a standardized and comprehensible manner.

The Software Requirements Document

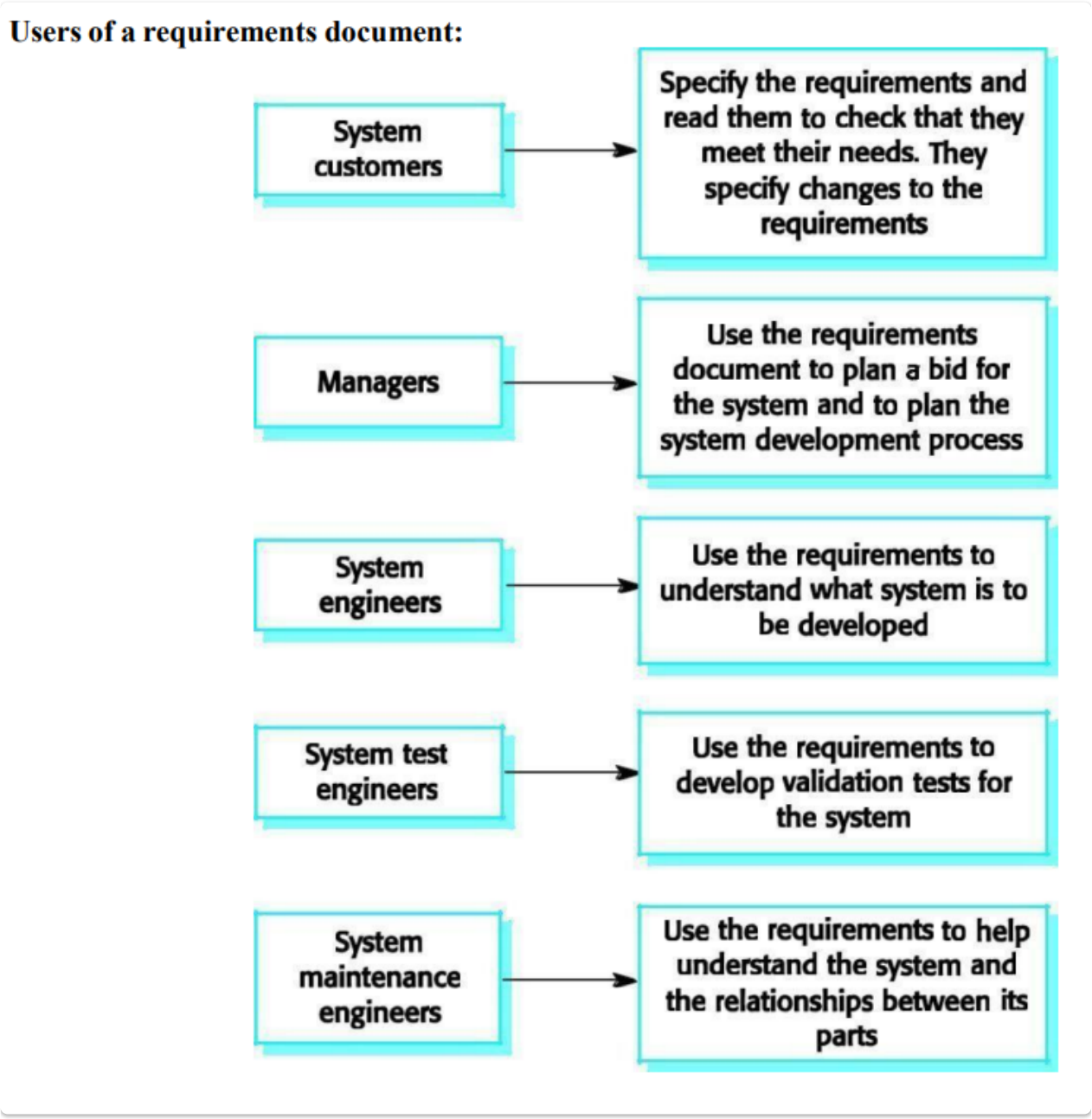
The Software Requirements Document

Software Requirements Document

The software requirements document serves as the official statement outlining what is expected from the system developers. It encompasses both the definition of user requirements and a detailed specification of system requirements. It is important to note that this document is not a design document; rather, it primarily focuses on WHAT the system should achieve rather than HOW it should achieve it.

Users of a Requirements Document

The IEEE requirements standard establishes a generic structure for a requirements document, which must be customized for each specific system. The document typically includes the following sections:



Structure

1. Introduction

- Purpose of the Requirements Document: Clearly articulates the goal and objectives of the document.
- Scope of the Project: Defines the boundaries and extent of the project.
- Definitions, Acronyms, and Abbreviations: Provides a glossary for understanding specific terms used.
- References: Lists any external documents or sources referenced in the requirements document.

- **Overview of the Remainder of the Document:** Gives a brief preview of the sections that follow.

2. General Description

- **Product Perspective:** Describes how the software product fits into the broader context, including dependencies on other systems.
- **Product Functions:** Enumerates the primary functions and features of the software.
- **User Characteristics:** Details the characteristics of the anticipated users.
- **General Constraints:** Outlines any overarching limitations or constraints.
- **Assumptions and Dependencies:** Specifies any assumptions made during the requirements analysis and identifies external dependencies.

3. Specific Requirements

This section comprehensively covers functional, non-functional, and interface requirements. It may include details on external interfaces, system functionality and performance, logical database requirements, design constraints, emergent system properties, and quality characteristics.

4. Appendices

5. Index

Requirements Engineering Process

Requirements Engineering Process

Requirements Engineering Process

Requirements Engineering Processes

The goal of the requirements engineering process is to create and maintain a comprehensive system requirements document. This process encompasses four high-level requirement engineering sub-processes, each addressing different aspects of the requirements:

1. Feasibility Study

- **Objective:** Assess whether the system is valuable and beneficial to the business.

- **Activities:** Evaluate the feasibility of implementing the proposed system, considering economic, technical, and operational aspects.

2. Elicitation and Analysis

- **Objective:** Discover and understand the requirements of the system.
- **Activities:** Engage stakeholders to gather information, analyze gathered data, and define both functional and non-functional requirements.

3. Specification

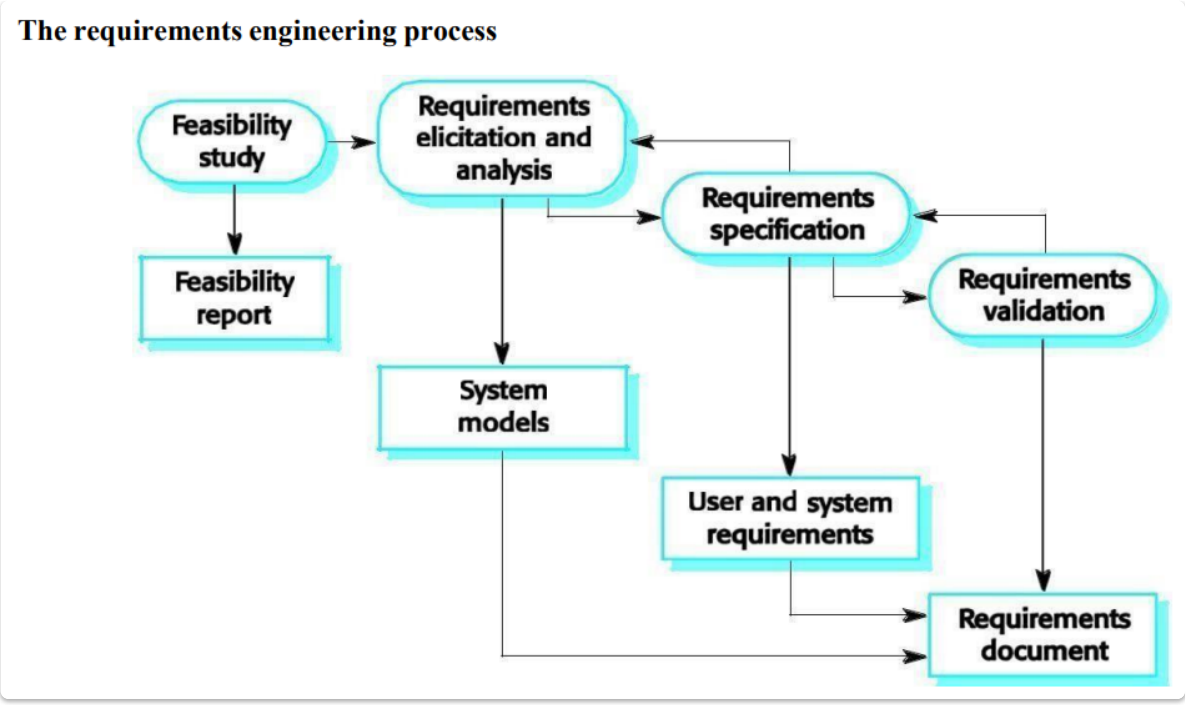
- **Objective:** Convert requirements into a standardized form for documentation.
- **Activities:** Clearly articulate and document the requirements, often using standardized templates and forms. This involves defining functional and non-functional aspects.

4. Validation

- **Objective:** Ensure that the documented requirements accurately reflect the system the customer desires.
- **Activities:** Verify the requirements by checking against the customer's needs and expectations, addressing any discrepancies or ambiguities.

Requirement Management

- **Objective:** Manage changes in requirements throughout the development process.
- **Activities:** Track, control, and document changes to requirements, ensuring that the evolving system aligns with the changing needs and priorities.



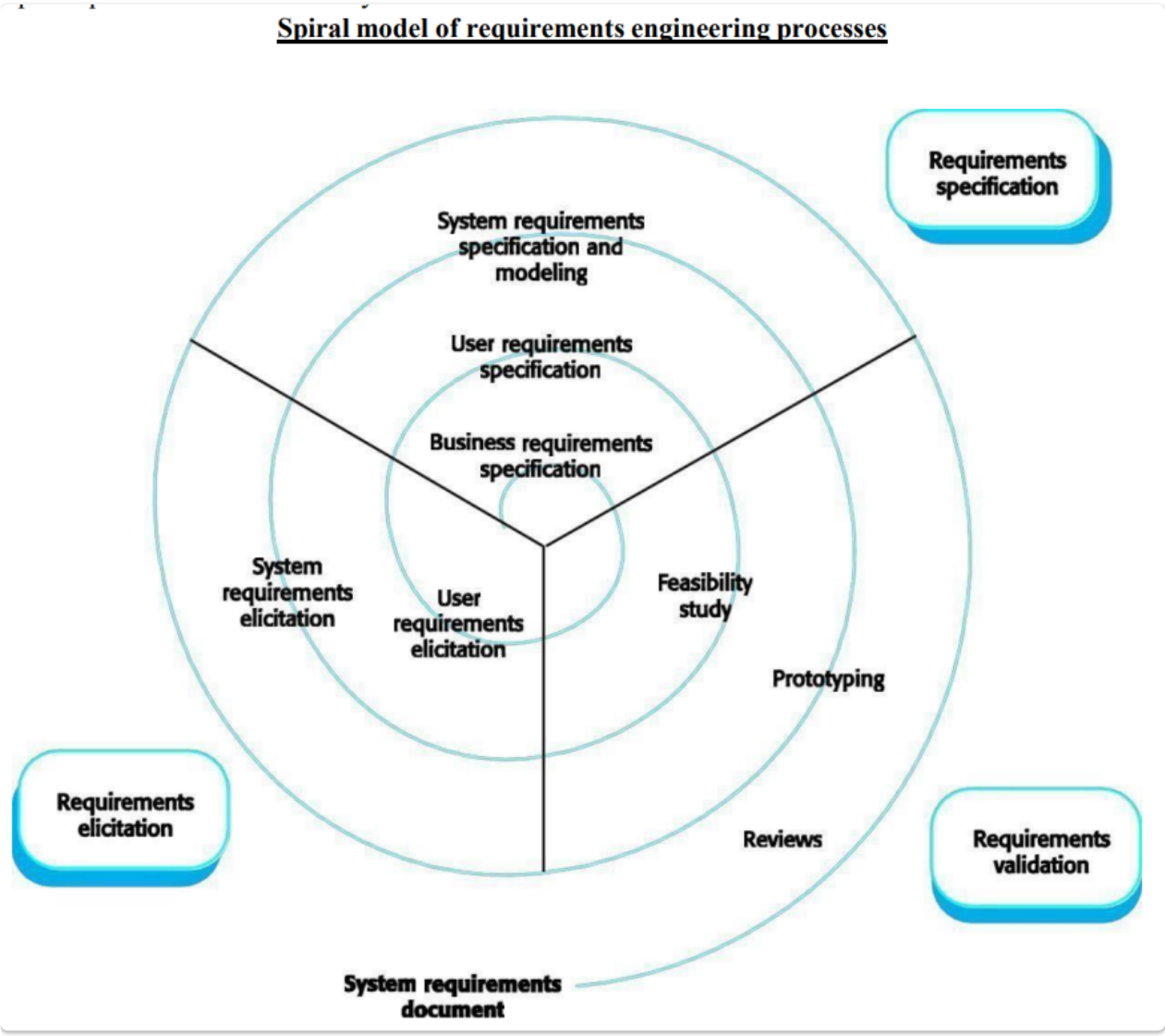
Alternative Perspective: The Spiral Model

An alternative perspective on the requirements engineering process presents it as a three-stage activity organized in an iterative manner around a spiral. The process involves understanding high-level business and non-functional requirements, eliciting user requirements, and progressing towards detailed system requirements and modeling. The spiral model accommodates varying levels of detail in requirements development, allowing for different iterations.

Structured Analysis Method

Some view requirements engineering as the application of a structured analysis method, such as object-oriented analysis. This entails analyzing the system, developing graphical system models (e.g., use-case models), and annotating them with additional information on performance or reliability. These models serve as a

detailed system specification.



Feasibility Studies

Feasibility Studies

A feasibility study is a critical phase that determines the viability of a proposed system. It aims to answer the fundamental question of whether or not the system is worthwhile. The input to the feasibility study includes preliminary business requirements, an outline description of the system, and its intended support for business processes. The study results in a report providing recommendations on whether to proceed with requirements engineering and system development.

Purpose of Feasibility Study

The feasibility study addresses three key aspects:

1. **Contribution to Organizational Objectives:**
    - Assess whether the proposed system aligns with and contributes to organizational objectives.

2. **Technological Feasibility:**
    - Evaluate if the system can be engineered using current technology and within budget constraints.
3. **Integration Feasibility:**
    - Determine if the system can be effectively integrated with other existing systems.

Feasibility Study Implementation

1. **Information Assessment:**
    - Assess the available information, including preliminary requirements and system description.
2. **Information Collection:**
    - Pose critical questions to stakeholders in the organization:
        - What if the system isn't implemented?
        - What are the current process problems?
        - How will the proposed system address these issues?
        - Anticipate integration challenges.
        - Identify technology needs and required skills.
        - Determine facilities supported by the proposed system.
3. **Information Sources:**
    - Consult various sources, including department managers, software engineers, technology experts, and end-users.
4. **Feasibility Study Timeline:**
    - Aim to complete the feasibility study within two to three weeks, gathering comprehensive insights.

Feasibility Study Report

1. **Recommendations:**
    - Provide a clear recommendation on whether to proceed with system development.
2. **Proposed Changes:**
    - Propose any necessary changes to the system's scope, budget, and schedule.
3. **High-Level Requirements:**
    - Suggest additional high-level requirements for the system based on the feasibility study findings.

The feasibility study is a focused effort to make informed decisions about the system's viability, considering organizational objectives, technological feasibility, and integration requirements. The resulting report guides stakeholders in determining the next steps in the system development process.

Requirements Elicitation and Analysis

Requirements Elicitation and Analysis

Requirements Elicitation and Analysis

The heart of the requirements engineering process is requirements elicitation and analysis, also known as requirements discovery. This phase involves technical staff collaborating with customers to understand the application domain, define the services the system should provide, and identify operational constraints. The participants in this process may include end-users, managers, maintenance engineers, domain experts, and other stakeholders.

Challenges in Requirements Analysis

1. **Unclear Stakeholder Desires:**
   - Stakeholders may not have a clear understanding of what they truly want from the system.
2. **Diverse Expression of Requirements:**
   - Stakeholders express requirements using their own terms and perspectives.
3. **Conflicting Stakeholder Requirements:**
   - Different stakeholders may have conflicting requirements, leading to challenges in prioritization.
4. **Influence of Organizational and Political Factors:**
   - Organizational and political factors can impact system requirements, adding complexity.
5. **Dynamic Nature of Requirements:**
   - Requirements may change throughout the analysis process due to emerging stakeholders or shifts in the business environment.

The Requirements Spiral

Process Activities

1. **Requirements Discovery:**

- Interact with stakeholders to understand and discover their requirements, including domain requirements.

2. **Requirements Classification and Organization:**
   - Group related requirements and organize them into coherent clusters.
3. **Prioritization and Negotiation:**
   - Prioritize requirements and resolve conflicts through negotiation with stakeholders.
4. **Requirements Documentation:**
   - Document requirements, providing input for the next iteration of the spiral.

Process Cycle

The requirements engineering cycle initiates with requirements discovery and concludes with requirements documentation. With each cycle, the analyst's comprehension of requirements improves.

Requirements Classification and Organization

This activity focuses on identifying overlapping requirements from various stakeholders and grouping related requirements. System architecture models are often used to associate requirements with specific subsystems.

Negotiation and Prioritization

Stakeholders may have diverse perspectives on the importance and priority of requirements, often leading to conflicts. Regular negotiation sessions are essential to reach compromises.

Requirement Documentation

In the documentation stage, requirements elicited are documented in a way that facilitates further requirements discovery and guides subsequent phases of the development process.

Requirements Validation

Requirements Validation

Requirements validation is a crucial step in the requirements engineering process, focused on demonstrating that the defined requirements align with the actual needs of the customer. Given the high costs associated with requirements errors,

validation becomes paramount. Fixing a requirements error after system delivery can be up to 100 times more expensive than addressing an implementation error.

Requirements Checking Criteria

1. **Validity:**
   - Does the system provide the functions that best support the customer's needs?
2. **Consistency:**
   - Are there any conflicts or contradictions among the requirements?
3. **Completeness:**
   - Are all functions required by the customer included in the specifications?
4. **Realism:**
   - Can the requirements be feasibly implemented given the available budget and technology?
5. **Verifiability:**
   - Can the requirements be effectively checked and tested?

Requirements Validation Techniques

1. Requirements Reviews

- **Description:**
  - Systematic manual analysis of the requirements.
- **Involvement:**
  - Both client and contractor staff should participate in reviews.
- **Types:**
  - Reviews can be formal (with completed documents) or informal.

2. Prototyping

- **Description:**
  - Using an executable model of the system to validate requirements.
- **Coverage:**
  - Detailed coverage in Chapter 17.

3. Test-Case Generation

- **Description:**
  - Developing tests for requirements to assess testability.

Requirements Reviews

- **Timing:**
  - Regular reviews during the formulation of requirements are crucial.
- **Participants:**
  - Involvement of both client and contractor staff.
- **Formality:**
  - Reviews can be formal (with completed documents) or informal.

Review Checks

1. **Verifiability:**
   - Is the requirement realistically testable?
2. **Comprehensibility:**
   - Is the requirement clearly understood by all stakeholders?
3. **Traceability:**
   - Is the origin of the requirement clearly stated and understood?
4. **Adaptability:**
   - Can the requirement be changed without causing significant impacts on other requirements?

Effective communication between developers, customers, and users during reviews is key to resolving problems at an early stage and ensuring a successful requirements validation process.

Requirements Management

Requirements Management

Requirements Management

Requirements management is a critical process in the requirements engineering and system development lifecycle. It involves handling changing requirements, which are inherently incomplete and often inconsistent due to evolving business needs and a developing understanding of the system.

Challenges in Requirements Management

1. **Incomplete and Inconsistent Requirements:**
   - New requirements emerge during the process, and different viewpoints may have conflicting requirements.
2. **Changing Priorities:**

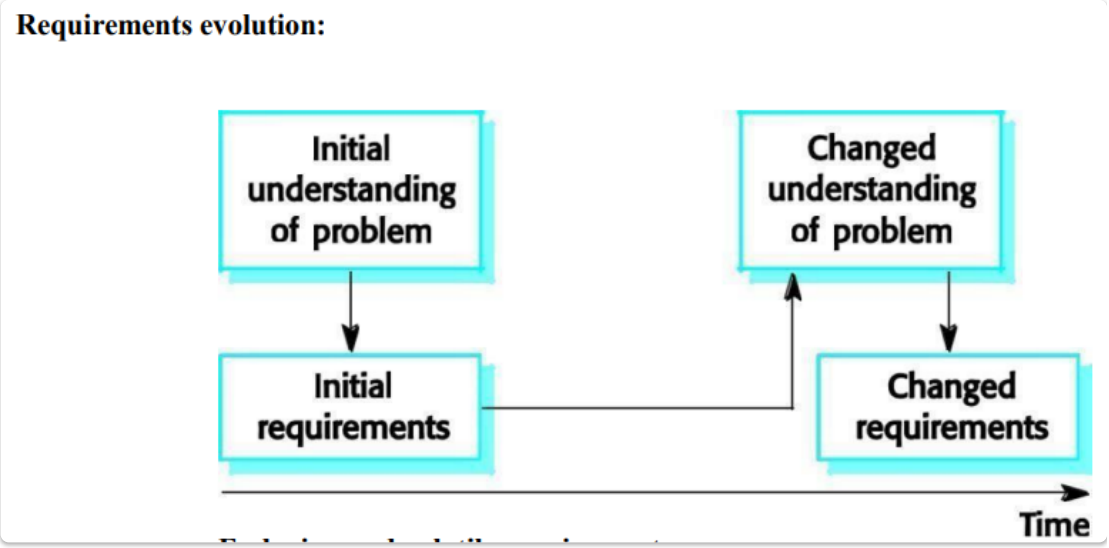
- The priority of requirements from different perspectives changes during development.

3. **Dynamic Business Environment:**
   - The business and technical environment of the system changes during its development.

Requirements Evolution

Enduring and Volatile Requirements

1. **Enduring Requirements:**
   - Stable requirements derived from the core activities of the customer organization.
   - Example: A hospital will always have doctors and nurses.
2. **Volatile Requirements:**
   - Requirements that change during development or system use.
   - Example: Requirements derived from changes in health-care policy.



Requirements Classification

Type	Description
Mutable	Changes due to environmental shifts.
Emergent	Develop during system development.
Consequential	Result from the introduction of the computer system.
Compatibility	Depend on specific systems or business processes.

Requirements Management Planning

During the requirements engineering process, planning includes:

1. **Requirements Identification:**
    - How requirements are individually identified.
2. **Change Management Process:**
    - The process followed when analyzing a requirements change.
3. **Traceability Policies:**
    - Amount of information about requirements relationships maintained.
4. **CASE Tool Support:**
    - Tool support required to manage requirements change.

Traceability

- **Source Traceability:**
    - Links from requirements to stakeholders who proposed them.
- **Requirements Traceability:**
    - Links between dependent requirements.
- **Design Traceability:**
    - Links from requirements to the design.

Requirements Change Management

Principal Stages:

1. **Problem Analysis:**
    - Discuss requirements problem and propose change.
2. **Change Analysis and Costing:**
    - Assess effects of change on other requirements.
3. **Change Implementation:**
    - Modify requirements document and other documents to reflect change.

Requirements change management should apply to all proposed changes, and the process involves analyzing, assessing impacts, and implementing modifications to the requirements document.

System Modeling

System modeling aids analysts in understanding system functionality and communicates effectively with customers. Different models offer various

perspectives, such as the behavioral and structural aspects of the system.

Model Types

1. **Data Processing Model:**
   - Illustrates how data is processed at different stages.
2. **Composition Model:**
   - Shows how entities are composed of other entities.
3. **Architectural Model:**
   - Reveals principal sub-systems.
4. **Classification Model:**
   - Demonstrates how entities share common characteristics.
5. **Stimulus/Response Model:**
   - Depicts the system's reaction to events.

Context Models

Context models illustrate the operational context of a system, showing what lies outside the system boundaries. Social and organizational considerations may influence decisions regarding system boundaries. Architectural models depict the system and its relationships with other systems.