Linnéuniversitetet

Software Design (20VT-2DV608)

Requirements Engineering

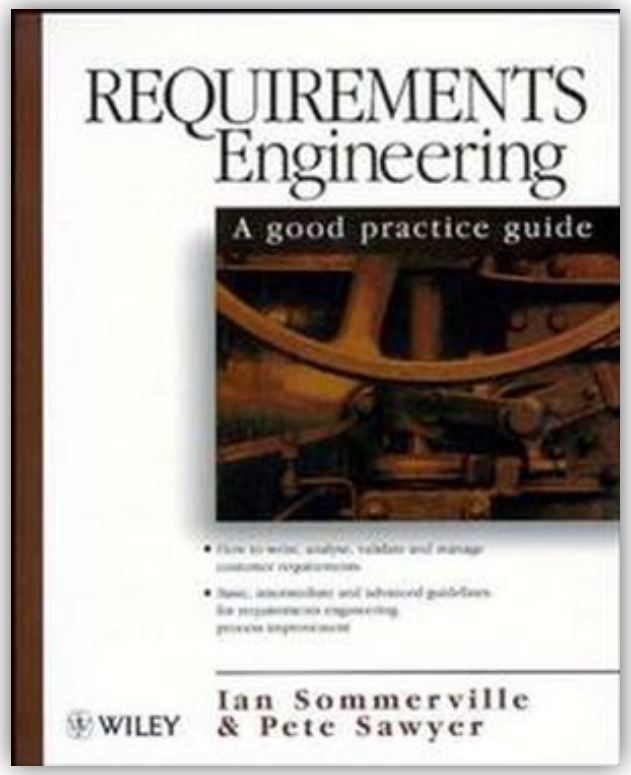
Francis Palma

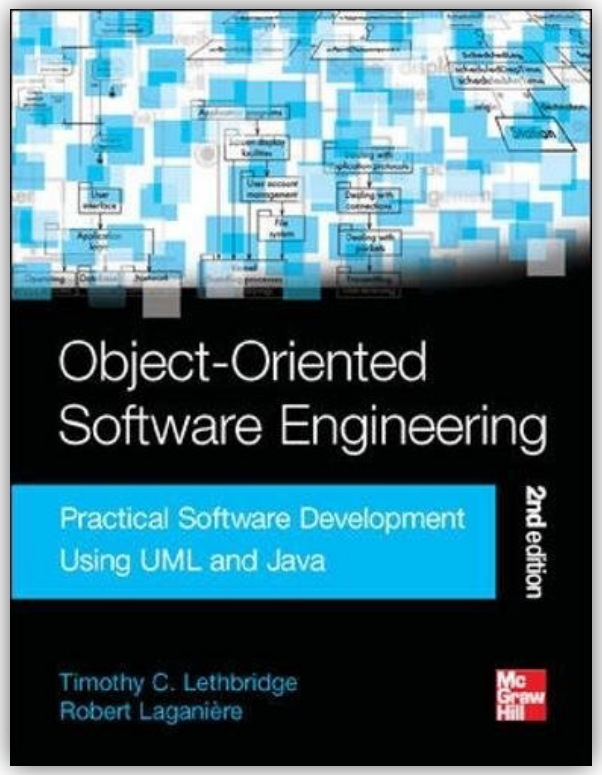
francis.palma@lnu.se

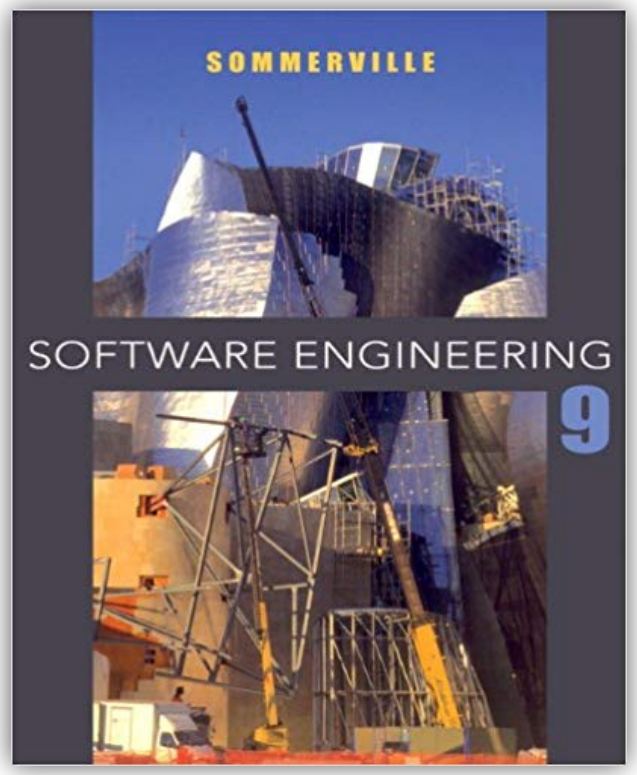
Department of Computer Science and Media Technology

Linnaeus University

Books and References







Course Outline

- Why Software Engineering - Design (Mauro Caporuscio)
  - w4
- Requirement Engineering (Francis Palma)
  - w5.1 (Jan 29th): Understanding and Elicitation of Software Requirements
  - w5.2 (Jan 31st): Requirements Validation and Management
  - w6.1 (Feb 5th): Modeling with UML
  - w6.2 (Feb 7th): Requirements Modelling and Management with Tools
  - W7: ASSIGNMENT RE
- Performance Engineering (Diego Perez)
  - w8.1 to w10
- Design and Refactoring (Mauro Caporuscio)
  - w11.1 to w13

Course Outline

- Why Software Engineering - Design (Mauro Caporuscio)
  - w4
- Requirement Engineering (Francis Palma)
  - **w5.1 (Jan 29th): Understanding and Elicitation of Software Requirements**
  - w5.2 (Jan 31st): Requirements Validation and Management
  - w6.1 (Feb 5th): Modeling with UML
  - w6.2 (Feb 7th): Requirements Modelling and Management with Tools
  - W7: ASSIGNMENT RE
- Performance Engineering (Diego Perez)
  - w8.1 to w10
- Design and Refactoring (Mauro Caporuscio)
  - w11.1 to w13

Understanding and Elicitation of Software Requirements

- Introduction to Requirements and Requirements Engineering
- Requirements Document
- Requirements Elicitation
- Requirements Analysis and Negotiation
- Describing Requirements

Understanding and Elicitation of Software Requirements

- Introduction to Requirements and Requirements Engineering
- Requirements Document
- Requirements Elicitation
- Requirements Analysis and Negotiation
- Describing Requirements

What are Software Requirements?

- Software requirements are descriptions of: how the system should behave, a system property or attribute. Defined during the early stages of system development.

- A requirement describes:
  - A user-level facility (e.g. 'word processor must include spell checking and correction command’)
  - A very general system property (e.g. 'the system must ensure that personal information is never made available without authorization’)
  - A specific constraint on the system (e.g. 'the sensor must be polled 10 times per second’)
  - A constraint on the development of the system (e.g. 'the system must be developed using Ada’)

- Ideally, requirements should be statements of ***what*** a system should do instead of ***how*** it should do it.

IEEE Definition of Software Requirements

- The *IEEE Standard Glossary of Software Engineering Terminology* (IEEE 610.12-1990) defines a requirement as:

  1. A condition or capability needed by a user to solve a problem or achieve an objective.

  2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.

  3. A documented representation of a condition or capability as in 1 or 2.

What is Requirements Engineering?

- "*Requirements engineering (RE) refers to the process of defining, documenting, and maintaining requirements in the engineering design process*." (Bashar Nuseibeh and Steve Easterbrook: *Requirements Engineering: A Roadmap,* 2000)
- The term *'engineering*' implies that systematic and repeatable techniques should be used to ensure that system requirements are complete, consistent, and relevant.
- In the waterfall model, RE is presented as the first phase of the development process.
  - Although, other development methods, e.g., Rational Unified Process, assume that RE continues through the lifetime of a system.

What is a Requirements Document?

- The *requirements document* is an official statement of the system requirements for customers, end-users, and software developers.

- The *requirements document* is also known as the *'functional specification*', the *'requirements definition*', the *'software requirements specification* (SRS)', etc.

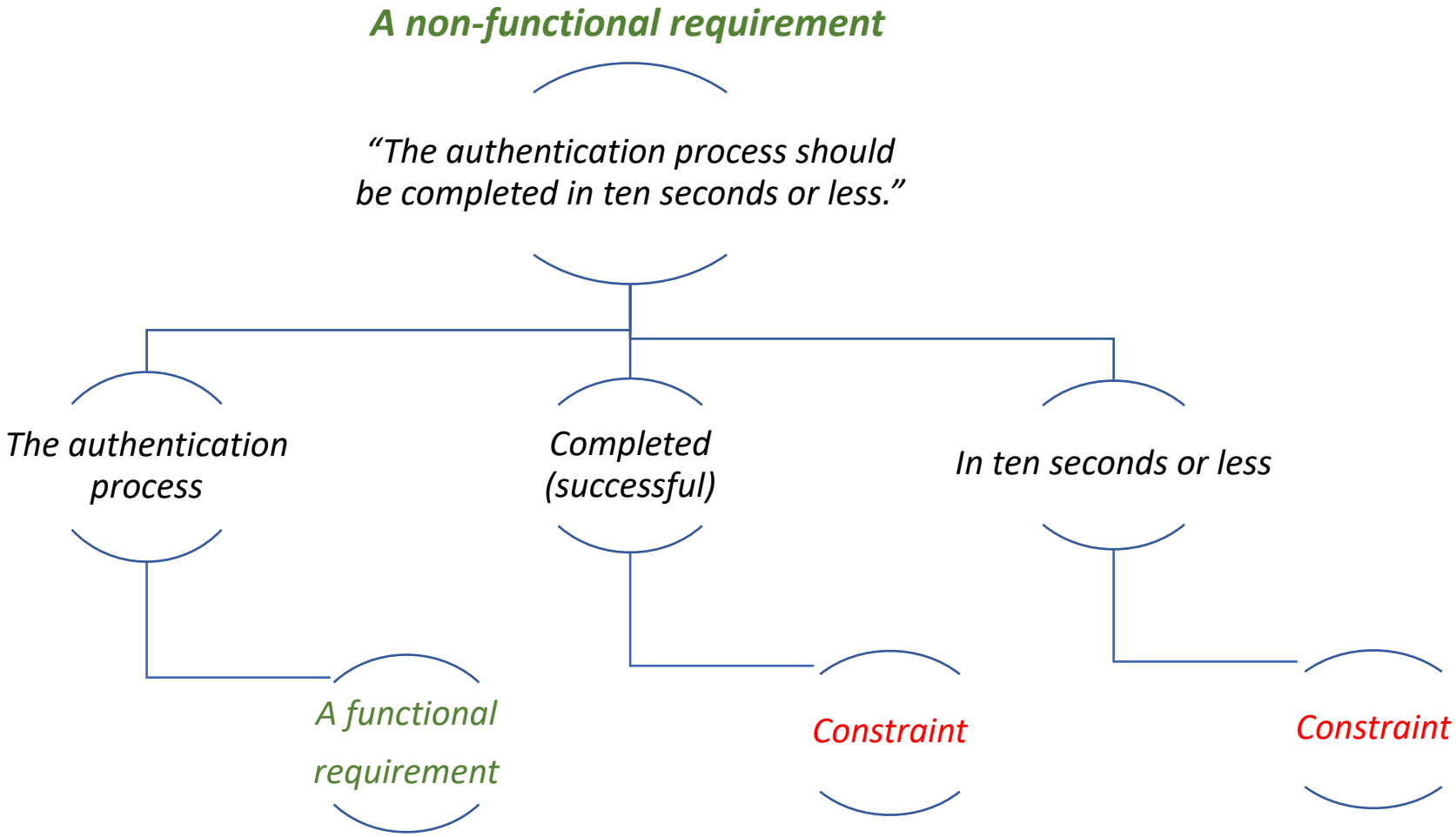
Types of Requirements: A Brief Idea

- On the basis of level of details:
  - **Stakeholder requirements** (or user requirements): not expressed in great detail and often described in natural language or informal diagrams.
  - **System requirements**: More detailed specifications of requirements expressed as an abstract model of the system. This may be a mathematical model or based on graphical notations such as data-flow diagrams, object class hierarchies, etc. annotated with natural language descriptions.

- On the basis of goals:
  - **Functional requirements (FRs)**: The FRs describe what the system should do.
    - Example: *A system must provide some facility for authenticating the identity of a system user;*
  - **Non-functional requirements (NFRs)**: The NFRs place constraints on how these functional requirements are implemented.
    - Example: *The authentication process should be completed in ten seconds or less.*

- *High-level non-functional requirements are often decomposed into functional system requirements.*

Functional and Non-functional Requirements



[NFR] = [FR] + [Constraints]

System Stakeholders

- System stakeholders are affected by the system and who have a direct/indirect influence on the system requirements. The system stakeholders include:
  - End-users of the system
  - Managers and others involved in the organizational processes
  - Engineers responsible for the system development and maintenance
  - Customers of the organization who will use the system to provide some services
  - External bodies such as regulators or certification authorities, etc.
- An example: to develop an *Automated Railway Signaling System*, who are the stakeholders?
  - Operators responsible for running the signaling system
  - Train crew
  - Railway managers
  - Passengers
  - Equipment installation and maintenance engineers
  - Safety certification authorities

- *As a good practice, we need to draw up an explicit list of stakeholders at an early stage in the RE process.*

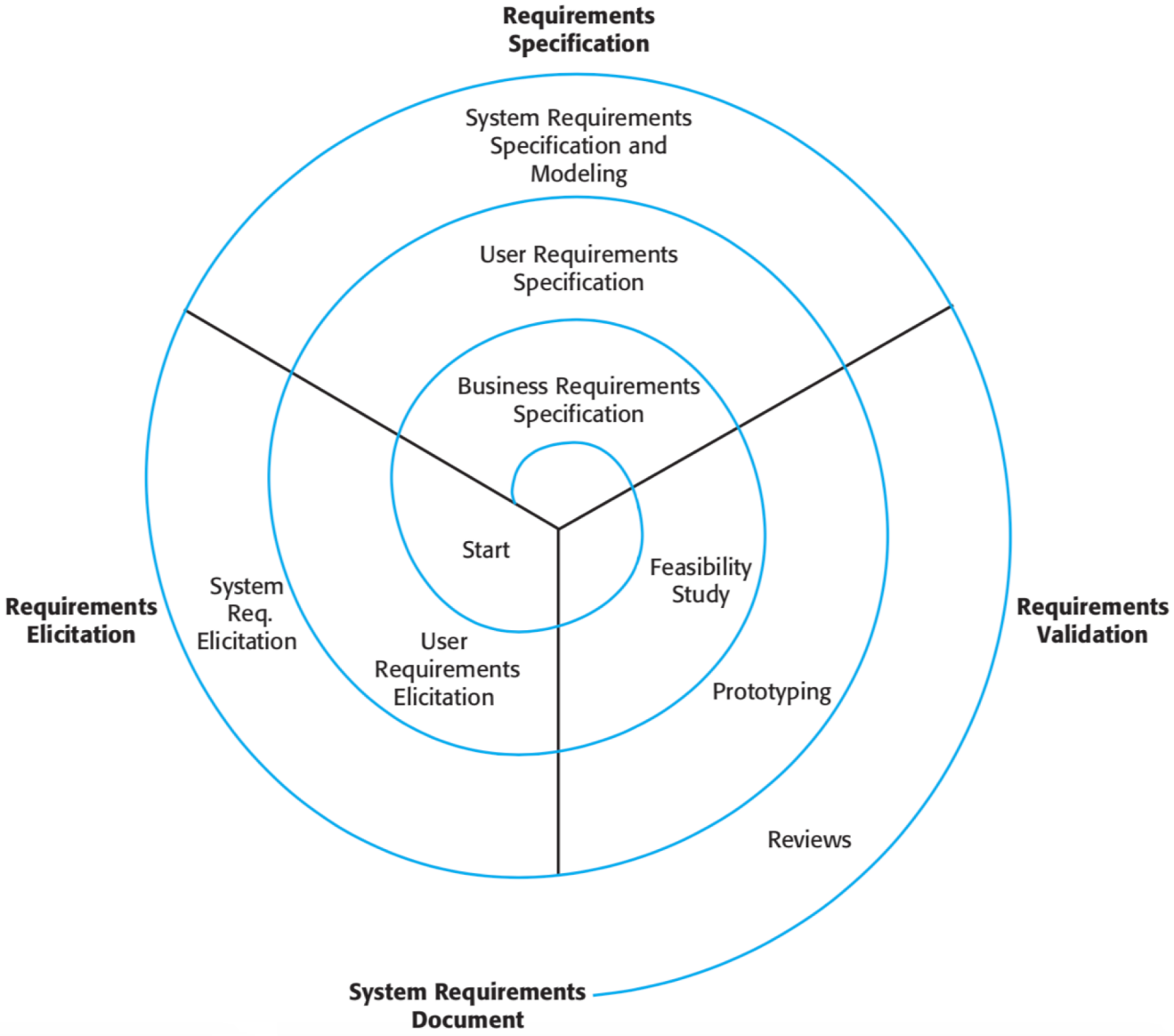
What is a Requirements Engineering (RE) Process?

- *"The RE process is a structured set of activities which are followed to derive, validate, and maintain a system's requirements document."*

- A complete process description should include:
  - What activities are carried out
  - The structuring or schedule of these activities
  - Who is responsible for each activity
  - The inputs and outputs to/from the activity
  - The tools used to support RE

- *In practice, a very few organizations explicitly define and standardize RE process. They simply define the end result of the process – the requirements document.*

Spiral View of Requirements Engineering Process



Problems in Requirements Engineering Process

- Two ways to recognize problems in RE process: through the information about the process or product.
- There is scope for RE process improvements, if you answer '***yes***' to some of the questions below:
  - Is your RE process usually over-budget and/or does it take longer than predicted?
  - Do people involved in RE complain that they do not have enough time or resources to do their job properly?
  - Are there complaints about the understandability or the completeness of the requirements documents?
  - Do system designers complain of rework resulting from requirements errors?
  - Do customers for your systems fail to use all of the system capabilities?
  - Is there a very high volume of change requests immediately after a system is delivered to customers?
- What if, you answer '***no***' to all the above questions?

What is a Good Requirements Engineering Process?

- There is no “one” good requirements engineering process!
  - Many possible ways to organize RE processes and they do not transfer well across organizations.

- Most of the standards developed for RE are concerned with process outputs such as the structure of requirements documents.
  - For example, the US DoD standard 2167A mentions RE activities but not a standard process.

- A good requirements engineering process include the following activities:
  - **Requirements Elicitation**: The system requirements are discovered through consultation with stakeholders, from system documents, domain knowledge, and market studies.
  - **Requirements Analysis and Negotiation**: The requirements are analyzed in detail. There are formal negotiation process involving different stakeholders to accept/reject requirements.
  - **Requirements Validation**: These should be a careful check of the requirements for consistency/completeness.

Standards for Requirements Engineering Process

- The *ISO/IEC/IEEE 90003:2018 Software Engineering* is a set of guidelines developed for organizations in the application of ISO 9001 to the acquisition, supply, development, operation, and maintenance of computer software and related support services.

- The *DOD-STD-2167A* (Department of Defense Standard 2167A) titled "*Defense Systems Software Development*" is a United States defense standard published in 1988.
  - Established "*uniform requirements for the software development that are applicable throughout the system life cycle".*

Understanding and Elicitation of Software Requirements

- Introduction to Requirements and Requirement Engineering
- Requirements Document
- Requirements Elicitation
- Requirements Analysis and Negotiation
- Describing Requirements

Requirements Document

- The requirements document is an official statement of the system requirements for customers, end-users, and software developers.

- Depending on the organization, the requirements document may have different names such as the *'functional specification*', *'the requirements definition*', *'the software requirements specification*', etc.

- Good quality requirements documents are:
  - Clear and consistent specification of the system to be implemented.
  - Readily understandable by a range of different readers.
  - As concise as possible and are designed so that changes can be performed easily and cheaply.

A Standard Document Structure

- The best document structure for your needs depends on custom and practice in your organization and the type of system being developed.

- Benefits of a standard format:
  - A standard format for requirements documents means that **readers** can use their knowledge of previous documents when reading a new requirements document, i.e., can find information more easily and understand the relationships between different parts of the document.
  - The standard format acts as a checklist for **writers** of requirements documents and reduces the chances for accidentally omitting information.
  - Document **reviewers** can use the standard (i) to check if sections have been left out of the document and (ii) as a driver of the reviewing process.

- One of the most accessible standards is the IEEE/ANSI 830-1993 standard.

The Structure of IEEE/ANSI 830-1993 Standard

1 Introduction

- 1.1 Purpose of the requirements document
- 1.2 Scope of the product
- 1.3 Definitions, acronyms, and abbreviations
- 1.4 References
- 1.5 Overview of the remainder of the document

2 General description

- 2.1 Product perspective
- 2.2 Product functions
- 2.3 User characteristics
- 2.4 General constraints
- 2.5 Assumptions and dependencies

3 Specific functional, non-functional, & interface requirements. Example: external interfaces, functionality, performance requirements, logical database requirements, design constraints, system attributes, and quality characteristics.

4 Appendices

5 Index

Explain ‘How to Use the Document’

- Include a section in the introduction of a document to explain how it should be used.
- Benefits include:
  - The **readers** will spend less time reading and understanding the system requirements.
  - If you inform readers what they need to know in order to understand a document it helps them judge if the problems of comprehensibility are a result of their lack of technical background or it is due to badly-expressed requirements.

- The ‘how to use the document’ section may include:

1. The different types of reader that the document is aimed at, i.e., the target audience.
2. The technical background required to understand the document. Any specific technical knowledge required for specific sections.
3. Pointers to overview sections to give a general understanding of the requirements before reading the detailed specification.
4. Sections of the document which are intended for a specific type of reader, and may be skipped by others.
5. The order dependencies, e.g., describe the order in which sections should be read.

Include a ‘Summary of the Requirements’

- Always include an overview section in the requirements document and summarize the purpose of the system and principal system requirements.

- Benefits include:
  - Easy for readers to comprehend system requirements if they have a broad picture.
  - Creating a summary focuses attention on the critical requirements and can help establish the requirements priorities.

- *Organize the summary on a per chapter basis so that the reader can see what requirements are presented in each chapter of the document.*

Best Practices for Writing Requirements Documents

1. Use wide margins so that text lines are not too long. Use section and sub-section headings with a consistent style.

2. Use *emphasis* sparingly and consistently. You can emphasize information by using a **bold** or *italic* font or by underlining it.

3. Use tables, bulleted or numbered lists to present sets of related information items rather than list them within a sentence.

4. Where you need to show a number of items of information with some stable and some variant parts, use a table to show commonalities and differences.

5. Separate equations from the text using white space and present them using a different font.

6. If you are describing a sequence of events or a sequential process, use diagrams to show the steps in the process.

7. Do not use complex diagrams. Complex diagrams become very confusing and you should normally have less than 12 elements in a diagram.

Best Practices for Writing Requirements Documents

8. Produce documents in loose-leaf binders rather than in bound versions. Users may then selectively replace parts of the document.

9. Many word processors allow the automatic insertion of change bars to indicate that text has been changed, if available, use this facility.

10. When writing documents, avoid references to other page numbers in the document, as these page numbers will change when the document is modified.

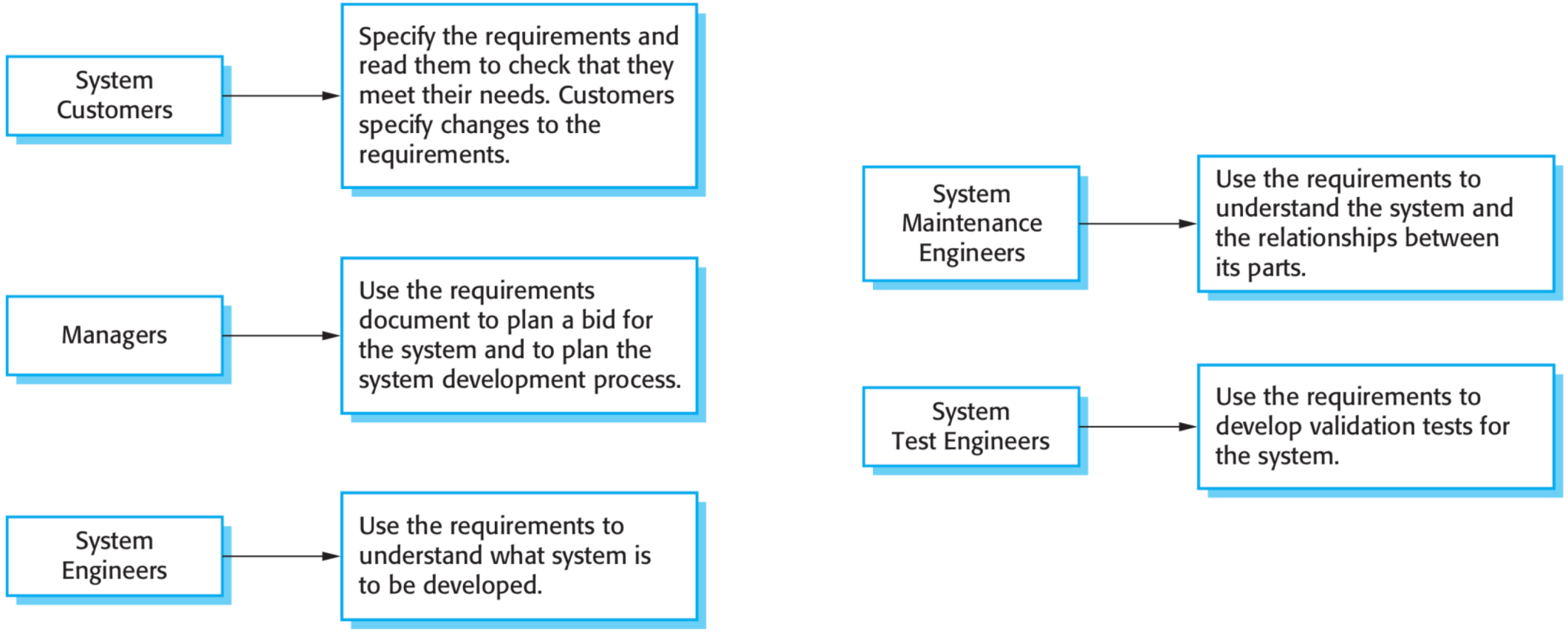
11. Ensure that all figures and tables are labelled and always refer to them by that label rather than, for example, 'the table below’.

12. Keep chapters short so that entire chapters may be replaced by document users.

13. Start new chapters on separate pages. This means that the chapter may be replaced without disrupting the remainder of the document.

14. Always number pages relative to chapters, i.e., the first page in chapter 2 is 2.1, the eighth page of chapter 12 is 12.8, etc.

Users of a Requirements Document




Linnéuniversitetet

Understanding and Elicitation of Software Requirements

- Introduction to Requirements and Requirement Engineering
- Requirements Document
- Requirements Elicitation
- Requirements Analysis and Negotiation
- Describing Requirements

What is Requirements Elicitation?

- The process of discovering the requirements for a system by communicating with customers, system users, and other who have a stake in the system development.

- It requires application domain and organizational knowledge as well as specific problem knowledge.

Identify and Consult Stakeholders

- A system stakeholder is anyone who benefits directly/indirectly from it, e.g., end-users, managers, customers, developers, etc.
  - All potential stakeholders need to be consulted to consider their specific requirements or constraints on the system.

- Benefits of identifying stakeholders:
  - If any stakeholder is missed, it is likely to miss important requirements;
  - Discussing the system with the identified stakeholders makes people feel that they are part of the requirements elicitation process.

- How to identify stakeholders?
  - Potential end-users of the system;
  - People involved in the business processes;
  - The customers of the organization who will use the system;
  - Personnel responsible for developing and maintaining the system;
  - External bodies, e.g., regulators/certification authorities who may place requirements on the system.

Define System's Operating Environment

- The operating environment of a system consists of the host computer, and other hardware and software systems which will interact with the system.
- Benefits:
  - You can anticipate and avoid many system installation problems that arise due to unexpected interactions with other installed systems.
  - Defining the environment helps reveal (i) system requirements for interaction with other systems, (ii) compatibility requirements where the new and existing systems have to run on the same platform.
- Key operating environment information:
  - **Platform information:** If the software is to run on an existing computer, you should specify the characteristics of that machine and the operating system which will be installed. Specify the libraries and versions of the operating system.
  - **Interface information:** If the system interacts directly with existing systems such as databases, discover and specify the interface provided by these system.
  - **Software dependencies:** The system may rely on other systems to provide functionality even it is not directly interfaced to them, e.g., the spreadsheet system.

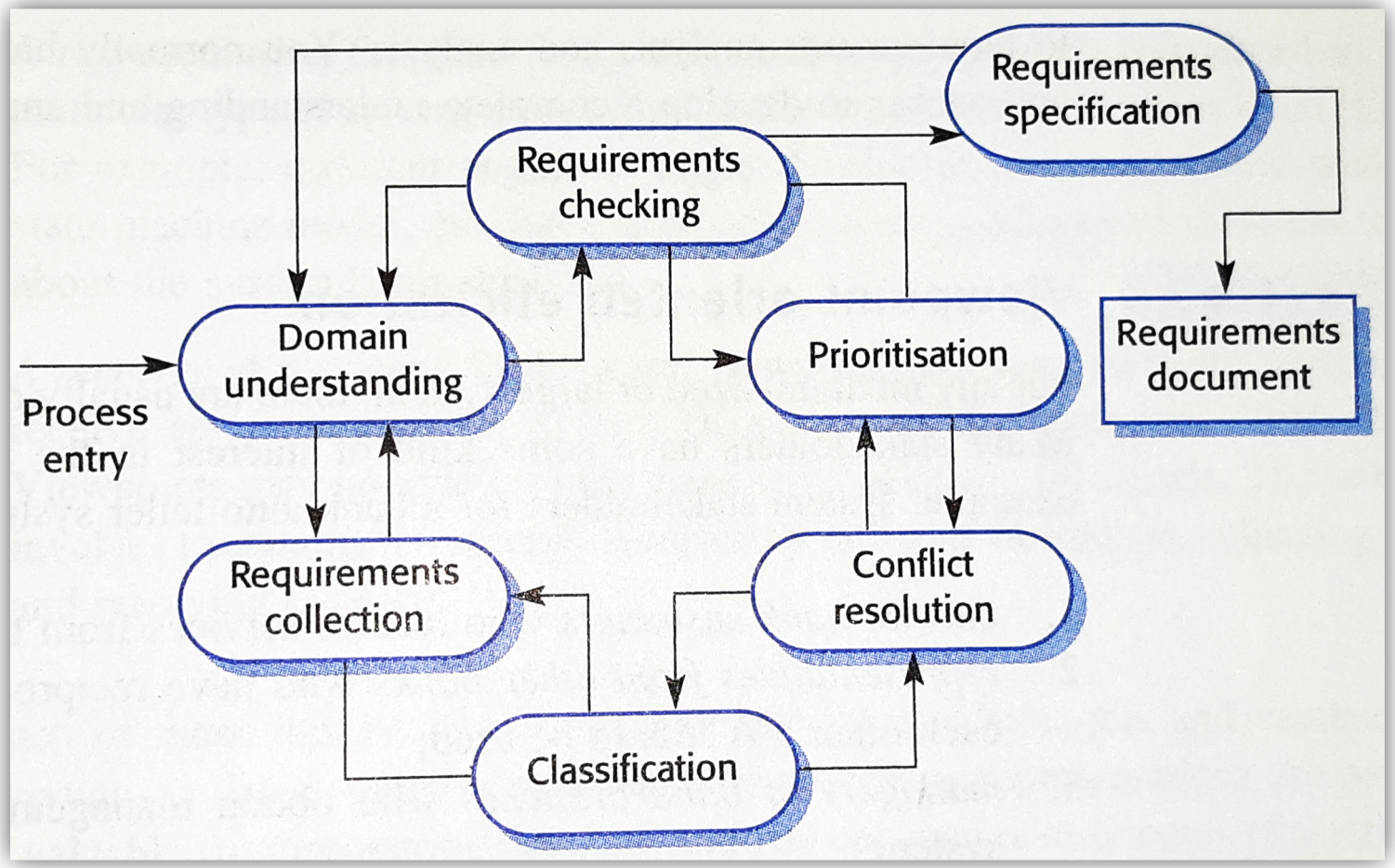
Scenarios-based Requirements Elicitation

- **Scenarios** are examples of interaction sessions between an end-user and the system.
  - End-users explain to the RE team what they are doing and the information they require from the system.
  - Scenarios can be seen as '***stories***' that explain how the system is used.
- Information that are crucial in scenarios:
  - A description of the state of the system before entering the scenario.
  - The normal flow of events in the scenario.
  - Exceptions to the normal flow of events.
  - Information about other activities which might be going on at the same time.
  - A description of the system after completion of the scenario.
- Usually 3-4 pages to describe each scenario, and should be supplemented with tables, flowcharts, etc.

Understanding and Elicitation of Software Requirements

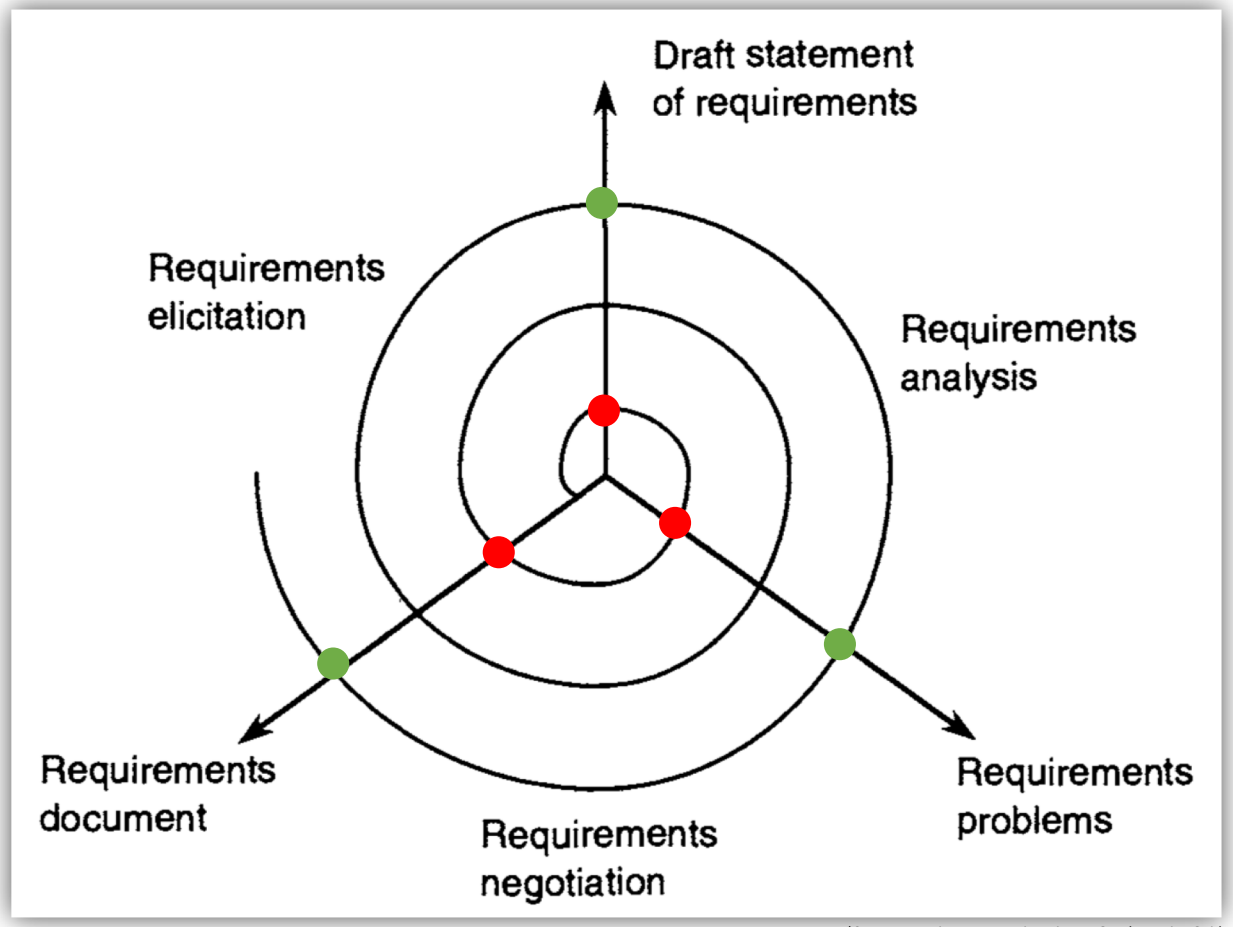
- Introduction to Requirements and Requirement Engineering
- Requirements Document
- Requirements Elicitation
- Requirements Analysis and Negotiation
- Describing Requirements

Requirements Elicitation and Analysis Process



The Elicitation, Analysis, and Negotiation Spiral

- **Requirements analysis** is an expensive and time-consuming process because skilled and experienced people must spend time reading documents carefully, and thinking about the implications of the statements in these documents.



Source: Requirements Engineering: A Good Practice Guide

Checklist for Requirements Analysis

- Develop checklists of requirements problems based on your experience, and use them in the systematic analysis of requirements.
- Analyze each requirement against the checklist and, if detected, note the problems.

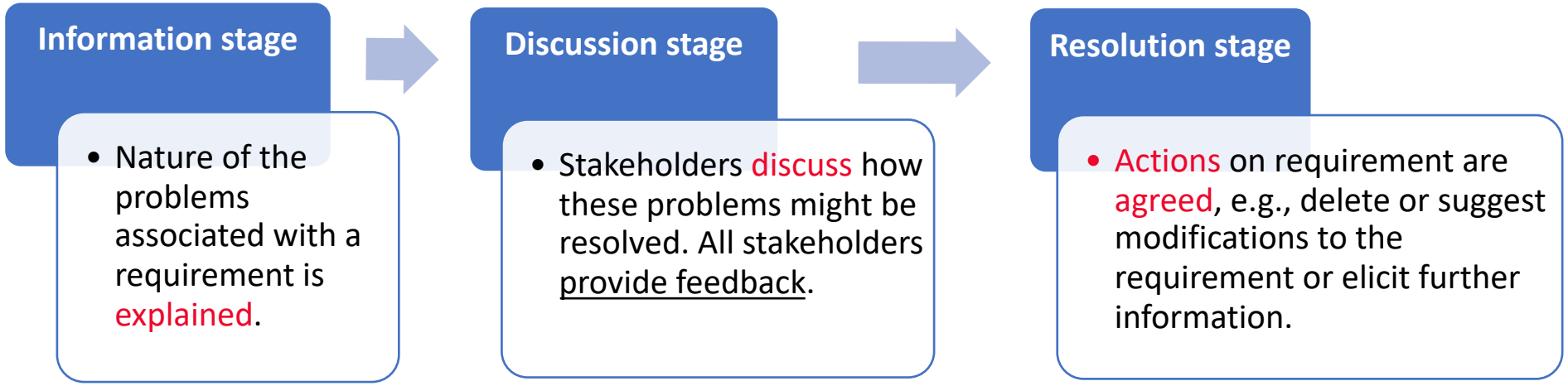
Premature design	Does the requirement include premature design or implementation information?
Combined requirements	Could the description of a requirement be broken down into several different requirements?
Unnecessary requirements	Is the requirement 'gold plating'? That is, a cosmetic addition to the system which is not really necessary.
Use of non-standard hardware	Does the requirement mean that non-standard hardware or software must be used?
Conformance with business goals	Is the requirement consistent with the business goals defined in the introduction to the requirements document?
Requirements ambiguity	Is the requirement ambiguous, i.e., could it be read in different ways by different people?
Requirements realism	Is the requirement realistic given the technology which will be used to implement the system?
Requirements testability	Is the requirement testable, that is, is it stated in such a way that test engineers can derive a test which can show if the system meets that requirement?

- Limit the size of checklists to about **10** questions.

Conflicts and their Resolution

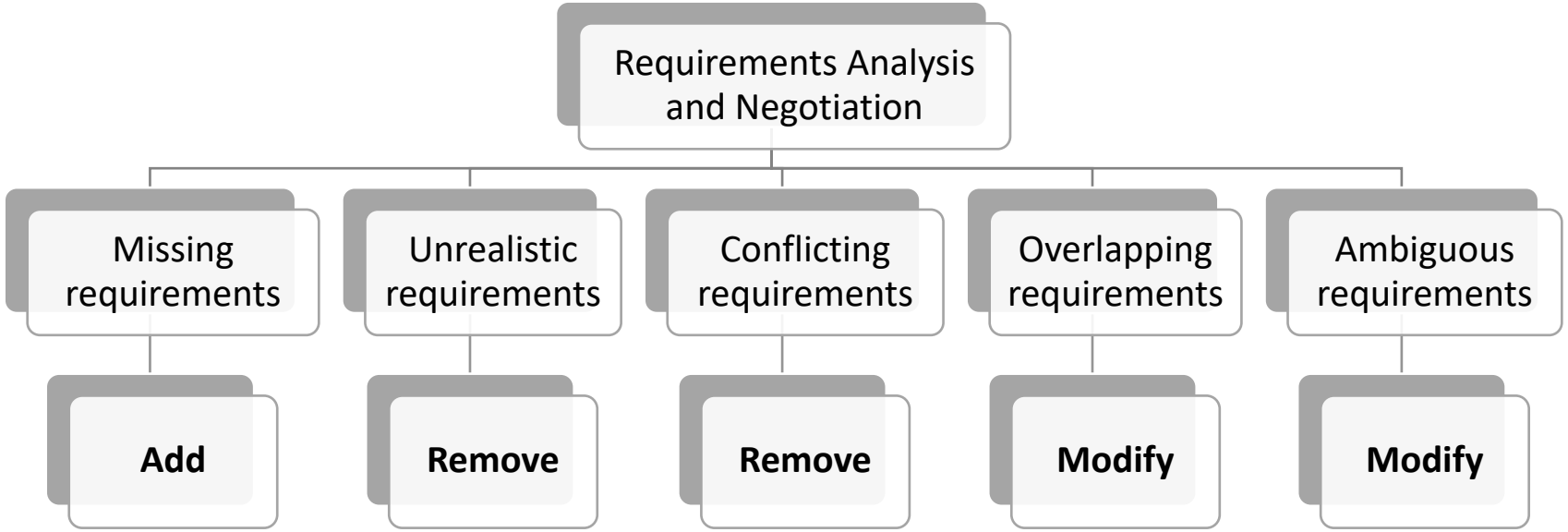
- There will always be conflicts, overlaps, and omissions in requirements. The requirements negotiation meetings can be arranged to discuss and resolve the problems discovered during the analysis.

- The negotiation meetings are conducted in three stages:



Requirements Analysis and Negotiation

- The goal is to establish an agreed set of requirements which are complete and consistent, not ambiguous.
- Requirements analysis and negotiation are concerned with the high-level statement of requirements elicited from stakeholders.
  - Requirements engineers and stakeholders negotiate to agree on a definition of the system requirements.



Prioritize Requirements

- Priorities should be assigned to requirements to reflect their importance to stakeholders and to the overall success of the system.
  - assigned in discussions between requirements analysts and stakeholders.
  - helps stakeholders to decide on the core requirements for the system.

- Ideally, priorities should be assigned during the elicitation process. However, it is difficult to assign priorities at this stage as they do not have a complete picture of the system requirements.

- Decide on a small number of priority classifications:
  - **Essential** - it must be included in the system;
  - **Useful** - the system will be less effective without it;
  - **Desirable** - it is not a core system facility but makes the system more attractive to users;

Classify Requirements

- The goal of classifying requirements is to relate requirements.
- Try have a multi-dimensional classification, i.e., requirements may fall into multiple classes.
- Benefits include:
  - Helps finding missing requirements, i.e., if there are no requirements in some classification scheme, then some important requirements may have been left out.
- The simplest way to classify requirements is to use what is called *a faceted approach*, i.e., identify a number of dimensions or facets and identify keywords which describe each of these.

After deciding the classification terms, read each requirement and associate one or more of these keywords.

System
- Requirements that affect the entire system such as performance or reliability requirements.

User interface
- Requirements that are concerned with user interaction.

Database
- Requirements that are concerned with the data managed by the system.

Communications
- Requirements that are concerned with the external communication facilities in the system.

- *'A Graphical User Interface to the Account Data Should be Provided'*. What are possible classifications?

'User interface' 'Database' 'Security'

Assess Requirements Risks

- For each requirement, perform risk analysis to identify possible problems that may arise in the implementation and the effects of those problems on the system.

Types of Risks in Implementing a Requirement

- *Performance risks*: May affect the overall performance of the system.
- *Safety and security risks*: May cause problems in meeting overall system requirements for safety and security.
- *Process risks*: May require changes to the normal development process, e.g., the introduction of mathematical specifications (for safety requirement) or the use of unfamiliar prototyping systems (for UI requirement).
- *Implementation technology risks*: May require the use of unfamiliar implementation technology, such as AI techniques, the use of N-version programming for fault tolerance, etc.
- *Database risks*: May involve non-standard data which is not available in an existing system database.
- *Schedule risks*: May be technically difficult and may threaten the planned development schedule for the system.
- *External risks*: Involves external contractors.
- *Stability risks*: Requirement may be volatile and subject to evolution during the development process.

- **Note:** *Risk assessment is imprecise – do not use a numeric assessment scheme for risks. Instead, do the risk assessment using **'fuzzy**' categories such as 'high', 'medium' or 'low'.*

Understanding and Elicitation of Software Requirements

- Introduction to Requirements and Requirement Engineering
- The Requirements Document
- Requirements Elicitation
- Requirements Analysis and Negotiation
- Describing Requirements

Describing Requirements

- In most organizations, system requirements are written as paragraphs of natural language, which is generally understandable but can be ambiguous, and is often misunderstood.
- Common problems if using natural language:
  - The requirements are written using complex conditional clauses (if A then if B then if C...) which are confusing.
  - Terminology is used in a sloppy and inconsistent way.
  - The writers of the requirement assume that the reader has specific knowledge of the domain or the system and they leave essential information out of the requirements document.
- Three essential things to improve requirements descriptions:
  - Investing effort in writing easy to read and understand requirements is always cost-effective.
  - Do not assume that all readers have the same background and knowledge.
  - Allow sufficient time to draft, review, and improve for requirements descriptions.

Ways of Writing Requirements Specification

Least Formal

Most Formal

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

Most Ambiguous

Least Ambiguous

General Guidelines for Describing Requirements

Define standard templates for describing requirements

- The template should include fields to collect the detailed information necessary to completely specify the requirement.
- Standard templates make requirements easier to collect, write, and read.
- If the requirement is a functional requirement, your standard may specify that the **inputs**, the **processing** and the **outputs** must be specified in the requirements definition.

Use language simply, consistently, and concisely

- While express a requirement in natural language, write descriptions using simple and concise language, avoiding complex sentence constructions, long sentences and paragraphs and ambiguous terminology.
- It takes less time to explain the requirements to stakeholders and a wider range of people can participate in the validation of the requirements.
- Generally, use **short sentences**, express one requirement in a single sentence, **avoid** using **jargons/abbreviations**, keep paragraphs short, use lists and tables, do not use nested conditional clauses to express requirements, use active rather than passive voice, etc.

General Guidelines for Describing Requirements

Use diagrams appropriately

- Use diagrams in a requirements description to show structural information or to show relationships between information in the requirements description.
- Diagrams break up sections of text into smaller, more readable fragments that can be reused, e.g., when making requirements presentations to customers.
- A good general rule for drawing diagrams is to keep them **as simple as possible**. Complex diagrams are often worse than no diagram at all.

Try to supplement natural language with other descriptions of requirements

- You can supplement requirements descriptions using specialized notations, such as **mathematical formula**, **decision tables**, design notations or programming languages, rather than only natural language.
- Specialized notations prevent misinterpretation than using a natural language description.
- Other models include: data-flow diagrams, timing diagrams, system models (e.g., object models, entity relation models, finite state models, etc.)

Key Takeaways

1. Requirements are descriptions of how the system should behave, or of a system property or attribute.

2. RE covers all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computer-based system.

3. The requirements document is an official statement of the system requirements for customers, end-users and software developers.

4. Types of requirements: User vs. System and Functional vs. Non-functional requirements.

5. The RE process is a structured set of activities which are followed to derive, validate, and maintain a systems requirements document.

6. Target audience of requirements documents: system customers, managers, system engineers, system test engineers, system maintenance staff, etc.

7. Requirements elicitation is the process of discovering the requirements for a system by communication with customers, system users and other who have a stake in the system development.

8. System stakeholder is anyone who benefits directly or indirectly from it, e.g., end-users, managers, customers, developers, etc.

9. The objective of requirements analysis and negotiation is to establish an agreed set of requirements which are complete and consistent, not ambiguous.

Linnéuniversitetet

Questions?