Certified Professional for Requirements Engineering

Foundation Level

Syllabus

Stan Bühne
Martin Glinz
Hans van Loenhoud
Stefan Staal

3.2.0 | February 26, 2024

Terms of Use

1. Individuals and training providers may use this syllabus as a basis for seminars, provided that the copyright is acknowledged and included in the seminar materials. Anyone using this syllabus in advertising needs the written consent of IREB e.V. for this purpose.
2. Any individual or group of individuals may use this syllabus as a basis for articles, books, or other derived publications, provided the copyright of the authors and IREB e.V. as the source and owner of this document is acknowledged in such publications.

Acknowledgements

This syllabus was initially created in 2007 by Karol Frühauf, Emmerich Fuchs, Martin Glinz, Rainer Grau, Colin Hood, Frank Houdek, Peter Hruschka, Barbara Paech, Klaus Pohl, and Chris Rupp. They were supported by Ian Alexander, Joseph Bruder, Samuel Fricker, Günter Halmans, Peter Jaeschke, Sven Krause, Steffen Lentz, Urte Pautz, Suzanne Robertson, Dirk Schüpferling, Johannes Staub, Thorsten Weyer, and Joy Beatty.

Version 3 is a major revision, created by Stan Bühne, Martin Glinz, Hans van Loenhoud, and Stefan Staal. They were supported by Karol Frühauf, Rainer Grau, Kim Lauenroth, Chris Rupp, and Camille Salinesi.

During this revision, feedback was provided by Xavier Franch, Karol Frühauf, Rainer Grau, Frank Houdek, and Thorsten Weyer. Supplementary feedback was provided by Wim Decoutere and Hans-Jörg Steffe.

Reviews were performed by Christoph Ebert, Barbara Paech, and Chris Rupp.

Approved for release on July 22, 2020 by the IREB Council upon recommendation of Xavier Franch and Frank Houdek.

We thank everybody for their involvement.

Preamble

In summer 2017, we conducted a survey to investigate the relevance of the existing Certified Professional for Requirements Engineering (CPRE) Foundation Level certification (version 2.2). The aim of the survey was to get feedback about the practical market relevance of this certification from the perspective of training providers as well as from certified CPRE practitioners working as Requirements Engineers [MFeA2019]. The survey reflected that the existing CPRE Foundation Level syllabus v2.2 generally still served the most important needs of the market and imparts the relevant RE knowledge to the candidates. Nevertheless, we

received feedback that several techniques were no longer used in practice, whereas others were missing in the search for more iterative and adaptive development. This feedback was in line with IREB's own perception of the changes in the field of Requirements Engineering (RE). We have therefore crafted a major revision of the CPRE Foundation Level syllabus, removing outdated content and adding new elements. Version 3 of the syllabus reflects the state of contemporary RE, covering both plan-driven and agile approaches to specifying and managing requirements.

Candidates seeking CPRE certification according to this syllabus are expected to have some basic knowledge of system development with plan-driven and agile approaches.

Purpose of the Document

This syllabus defines the educational objectives and a summary of the educational content for the Foundation Level of the certification Certified Professional for Requirements Engineering established by the International Requirements Engineering Board (IREB). The syllabus provides training providers with the basis for creating their course materials. Students can use the syllabus to prepare themselves for the examination.

Contents of the Syllabus

The Foundation Level addresses the needs of all people involved in the topic of Requirements Engineering. This includes people in roles such as Requirements Engineer, business analyst, system analyst, product owner or product manager, developer, project or IT manager, or domain expert.

This syllabus and the related handbook use the abbreviation "RE" for Requirements Engineering.

Content Scope

The CPRE Foundation Level communicates basic principles that are equally valid for any type of system (e.g., mobile apps, information systems, or cyber-physical systems). Furthermore, the CPRE Foundation Level does not assume any specific development process, nor is it geared to a specific application domain. Training providers may offer training that focuses on specific types of systems, processes, or application domains, as long as the educational objectives of this syllabus are fully covered.

Level of Detail

The level of detail in this syllabus allows teaching and examination to be consistent internationally. To achieve this goal, the syllabus contains the following:

- General educational objectives
- Contents with a description of the educational objectives
- References to further literature (where necessary)

Educational Objectives/Cognitive Knowledge Levels

All modules and educational objectives in this syllabus are assigned a cognitive level. The levels are classified as follows:

- **L1: Know** (describe, enumerate, characterize, recognize, name, remember, ...) — remember or retrieve previously learned material.
- **L2: Understand** (explain, interpret, complete, summarize, justify, classify, compare, ...) — grasp/construct meaning from given material or situations.
- **L3: Apply** (specify, write, design, develop, implement, ...) — apply knowledge and skills in given situations.

Higher levels include the lower ones. Note that all terms in the glossary which are designated as foundational terms must be known (L1), even if they are not explicitly mentioned in the educational objectives. The glossary is available for download on the IREB homepage at https://www.ireb.org/downloads/#cpre-glossary

Structure of the Syllabus

The syllabus consists of seven main chapters. Each chapter covers one educational unit (EU). The main chapter titles contain the cognitive level of their chapters, which is the highest level of their sub-chapters. The duration indicates the time that a training course should invest for that chapter. Training companies are free to allocate more time but should maintain the proportions between the EUs. Important terms used in a chapter are listed at the beginning of the chapter.

Example

EU 4	Practices for Requirements Elaboration (L3)
Duration:	4 hours 30 minutes
Terms:	Requirements source, system boundary, system context, requirements elicitation, requirements validation, stakeholder, Kano model, conflict, conflict resolution

This example shows that Chapter 4 contains educational objectives at level L3 and four and a half hours are intended for teaching the material in this chapter.

Each chapter contains sub-chapters. Their titles also contain the cognitive level of their content.

Educational objectives (EO) are enumerated before the actual text. The numbering shows the sub-chapter that they belong to. For example, the educational objective EO 4.2.1 is described in sub-chapter 4.2.

Order of Topics in the Syllabus

The order of chapters in this syllabus constitutes a logical order of topics. However, the topics do not have to be taught in exactly this order. Training providers are free to teach the

material in any order (including interleaving of topics from different EUs) that they deem appropriate in the context of their training and that fits their didactic concepts.

The Examination

This syllabus is the basis for the examination for the CPRE Foundation Level certificate.

A question in the examination can cover material from several chapters of the syllabus. All chapters (EU 1 to EU 7) of the syllabus can be examined.

The format of the examination is multiple choice.

Examinations can be held immediately after a training course but also independently of courses (e.g., in an examination center). A list of IREB licensed certification bodies can be found on the website https://www.ireb.org.

Version History

Version	Date	Comment
1.0	2007	First version
2.1	November 2010	First major update
2.2	March 1, 2015	Minor update (Removed inconsistencies, typos and grammar mistakes)
3.0.0	October 1, 2020	Second major update to reflect the state of contemporary RE, covering both plan-driven and agile approaches to specifying and managing requirements.
3.0.1	October 7, 2020	Fixed typos and renamed EU 4.3
3.1.0	September 1, 2022	Fixed typos and references in the whole document to improve readability. EU 1: Moved learning objective EO 1.3.2 to EO 1.1.2. Updated EO 1.2.2 and 1.3.1 EU 3 Updated education objective EO 3.1.3 EU 3.1.2: Reworked the paragraph on abstraction levels EU 3.4: All model types that do not need to be applied at Foundation Level have been moved to a new subsection 3.4.6 EU 3.6: Updated heading to "Requirements Documents and Documentation Structures" EU 4 EU 4.1: The description of stakeholders and stakeholder roles was made more precise. EU 4.2: The introduction and justification for design and idea generating techniques, has been updated to be more precise. EU 4.3: Updated heading to "Resolving Conflicts regarding Requirements" EU 6 EO 6.3.1 and EO 6.5.2 have been slightly modified

3.1.1	January 1, 2024	New Corporate Design implemented
3.2.0	February 26, 2024	Layout corrections for the new corporate design

Content

1 Introduction and Overview of Requirements Engineering (L2)

Goal: Know what RE is about and understand the value of RE

Duration: 1 hour

Terms: Requirement, requirements specification, Requirements Engineering (RE), stakeholder, system, Requirements Engineer

Educational objectives

EO 1.1.1 Remember the fundamental terminology (L1)
EO 1.1.2 Understand the different kinds of requirements (L2)
EO 1.2.1 Explain the value of RE (L2)
EO 1.2.2 Enumerate symptoms of inadequate RE (L1)
EO 1.3.1 Know where RE can be applied and where requirements occur (L1)
EO 1.4.1 Know the major tasks of RE and that an RE process has to be tailored for performing them (L1)
EO 1.5.1 Characterize the role and tasks of a Requirements Engineer (L1)
EO 1.6.1 Remember what a Requirements Engineer needs to learn (L1)

1.1 Requirements Engineering: What (L1)

People and organizations have desires and needs for new things to be built or existing things to be evolved. We call such needs *requirements*.

Things to be built or evolved may be:

- *Products* provided to customers
- *Services* made available to customers
- Any other *deliverables* such as devices, procedures, or tools that help people and organizations achieve a specific goal
- *Compositions* or *components* of products, services, or other deliverables

All these things can be considered as *systems*. In this syllabus, we use the term *system* to denote all kinds of things that *stakeholders* have requirements for. *Stakeholders* are persons or organizations who influence the requirements for a system or who are impacted by that system.

The goal of RE is to specify and manage requirements for systems such that the systems implemented and deployed satisfy their stakeholders' desires and needs.

In RE, we distinguish between three kinds of requirements [Glin2020]:

- *Functional requirements* concern a result or behavior that shall be provided by a function of a system. This includes requirements for data or the interaction of a system with its environment.
- *Quality requirements* pertain to quality concerns that are not covered by functional requirements, such as performance, availability, security, or reliability.

- *Constraints* are requirements that limit the solution space beyond what is necessary to meet the given functional requirements and quality requirements.

1.2 Requirements Engineering: Why (L2)

Adequate RE adds *value* in the process of developing and evolving a system:

- Reducing the risk of developing the wrong system
- Better understanding of the problem
- Basis for estimating development effort and cost
- Prerequisite for testing the system

Typical symptoms of inadequate RE are missing, unclear, or incorrect requirements. This is particularly due to:

- Rushing straight into building the system
- Communication problems between involved parties
- The assumption that the requirements are self-evident
- Inadequate RE education and skills

1.3 Requirements Engineering: Where (L2)

RE can be applied to requirements for any kind of system. However, the dominant application case for RE today is represented by systems in which software plays a major role. Such systems typically consist of software components, physical elements, and organizational elements.

Requirements can occur as:

- *System requirements* — what a system shall do
- *Stakeholder requirements* — what stakeholders want from their perspective
- *User requirements* — what users want from their perspective
- *Domain requirements* — required domain properties
- *Business requirements* — business goals, objectives, and needs of an organization

1.4 Requirements Engineering: How (L1)

The major tasks in RE are elicitation (4.2), documentation (3), validation (4.4), and management (6) of requirements. Tool support (7) may help perform these tasks. Requirements analysis and requirements conflict resolution (4.3) are considered to be part of elicitation. In order to perform RE activities properly, a suitable RE process must be tailored from a broad range of possibilities (5).

1.5 The Role and Tasks of a Requirements Engineer (L1)

Requirements Engineer typically is not a job title, but a *role* that people play who:

- Elicit, document, validate and/or manage requirements as part of their duties.
- Have in-depth knowledge of RE.
- Can bridge the gap between the problem and potential solutions.

In practice, business analysts, application specialists, product owners, systems engineers, and even developers act in the role of a Requirements Engineer.

1.6 What to Learn about Requirements Engineering (L1)

This syllabus covers the foundational skill set that a Requirements Engineer must learn. It encompasses the fundamental principles of RE (2), how to document requirements in various forms (3), how to elaborate requirements with various practices (4), how to define and work with suitable RE processes (5), how to manage existing requirements (6), and how to employ tool support (7).

2 Fundamental Principles of Requirements Engineering (L2)

Goal: Know and understand the principles of RE

Duration: 1 hour 30 minutes

Terms: Context, requirement, Requirements Engineering (RE), stakeholder, shared understanding, validation

Educational objectives

EO 2.1.1 Enumerate the principles of RE (L1)
EO 2.2.1 Remember the terms associated with the principles (L1)
EO 2.2.2 Explain the principles and the reasons why they are important (L2)

2.1 Overview of Principles (L1)

RE is governed by a set of fundamental principles which apply to all tasks, activities, and practices in RE. The following nine principles form the basis for the practices presented in the subsequent chapters of this syllabus.

1. Value-orientation: Requirements are a means to an end, not an end in itself
2. Stakeholders: RE is about satisfying the stakeholders' desires and needs
3. Shared understanding: Successful systems development is impossible without a common basis
4. Context: Systems cannot be understood in isolation
5. Problem – Requirement – Solution: An inevitably intertwined triple
6. Validation: Non-validated requirements are useless
7. Evolution: Changing requirements are no accident, but the normal case
8. Innovation: More of the same is not enough
9. Systematic and disciplined work: We can't do without in RE

2.2 The Principles Explained (L2)

Principle 1 – Value-orientation: Requirements are a means to an end, not an end in itself

The value of a requirement is equal to its benefit minus the cost for eliciting, documenting, validating, and managing the requirement. The benefit of a requirement is the degree to which it contributes to:

- Build systems which satisfy the desires and needs of its stakeholders.
- Reduce the risk of failure and costly rework when developing the system.

Principle 2 – Stakeholders: RE is about satisfying the stakeholders' desires and needs

As RE is about understanding the stakeholders' desires and needs, properly dealing with stakeholders is a core task of RE. Every stakeholder has a role in the context of the system to be built — for example, user, client, customer, operator, or regulator. A stakeholder may also have more than one role at the same time. For stakeholder roles with too many individuals or when individuals are unknown, fictional archetypical descriptions known as *personas* can be defined as substitute. It is not sufficient to consider only the requirements of end users or customers. Doing this would mean that we might miss critical requirements from other stakeholders. Users who provide feedback about a system in use should also be considered as stakeholders.

Stakeholders may have different needs and viewpoints, which may result in conflicting requirements. It is a task of RE to identify and resolve such conflicts.

Involving the right people in the relevant stakeholder roles is crucial for successful RE. Practices for identifying, prioritizing, and working with stakeholders are described in 4.

Principle 3 – Shared understanding: Successful systems development is impossible without a common basis

RE creates, fosters, and secures shared understanding between and among the parties involved: stakeholders, Requirements Engineers, and developers. There are two forms of shared understanding:

- *Explicit shared understanding* (achieved by documented and agreed requirements)
- *Implicit shared understanding* (based on shared knowledge about needs, visions, context, etc.)

Domain knowledge, previous successful collaboration, shared culture and values, and mutual trust are enablers for shared understanding, while geographic distance, outsourcing, or large teams with a high turnover are obstacles.

Proven practices for achieving shared understanding include the creation of glossaries (3.5), creating prototypes (3.7), or using an existing system as a reference point. Practices for assessing shared understanding include examples of expected outcomes, exploring prototypes, or estimating the cost of implementing a requirement. The most important practice for reducing the impact of misunderstandings is using a process with short feedback loops (5).

Principle 4 – Context: Systems cannot be understood in isolation

Systems are embedded in a *context*. Without understanding that context, it is impossible to specify a system properly. In RE, the context of a system is the part of a system's environment that is relevant for understanding the system and its requirements. The *system boundary* is the boundary between a system and its surrounding context. Initially, the system boundary is often not clear, and it may even change over time.

Clarifying the system boundary and defining the external interfaces between the system and the context elements it interacts with are genuine RE tasks. At the same time, the *scope* of the system—that is, the range of things that can be shaped and designed when developing the system—needs to be determined. We also need to consider the so-called *context boundary* which separates the RE-relevant part of the environment of a system from the rest of the world.

When specifying a system, it is not sufficient to consider only the requirements within the system boundary. In RE one must also consider:

- Changes in the context that may impact the system requirements.
- Real-world requirements relevant for the system (and how to map them to system requirements).
- Context assumptions that must hold for making the system work and meeting real-world requirements.

Principle 5 – Problem – Requirement – Solution: An inevitably intertwined triple

A *problem* occurs when stakeholders are not satisfied with the situation as is. *Requirements* capture what stakeholders need in order to get rid of the problem or to simplify it. A socio-technical system that satisfies these requirements constitutes a *solution*.

Problems, requirements, and solutions do not necessarily occur in this order. Solution ideas may create user needs which have to be worked out as requirements and implemented in an actual solution. This is typically the case when innovating.

- Problems, requirements, and solutions are closely intertwined: they cannot be tackled in isolation.
- Nevertheless, Requirements Engineers aim to separate problems, requirements, and solutions from each other as far as possible when thinking, communicating, and documenting. This separation of concerns makes RE tasks easier to handle.

Principle 6 – Validation: Non-validated requirements are useless

Eventually, we have to validate that the deployed system satisfies the stakeholders' desires and needs. In order to control the risk of unsatisfied stakeholders from the beginning, validation of requirements must start already during RE. We have to check whether:

- Agreement about the requirements has been achieved among the stakeholders,
- The stakeholders' desires and needs are adequately covered by the requirements,
- The context assumptions (see Principle 4 above) are reasonable.

Practices for validating requirements are discussed in 4.4.

Principle 7 – Evolution: Changing requirements are no accident, but the normal case

Systems and their requirements are subject to *evolution*. This means that they constantly change. Requests to change a requirement or a set of requirements for a system may be caused, for example, by:

- Changed business processes
- Competitors launching new products or services
- Clients changing their priorities or opinions
- Changes in technology
- Changes of laws or regulations
- Feedback from system users asking for new or changed features

Furthermore, requirements may change due to feedback from stakeholders when validating requirements, due to the detection of faults in previously elicited requirements, or due to changed stakeholder's needs.

As a consequence, Requirements Engineers must pursue two seemingly contradictory goals:

- Permit requirements to change
- Keep requirements stable

Details about how to achieve this are discussed in 6.7.

Principle 8 – Innovation: More of the same is not enough

Giving the stakeholders exactly what they want misses the opportunity to build systems that satisfy the stakeholders' needs better than they expect. Good RE strives not just to satisfy stakeholders, but to make them happy, excited, or feel safe. This is what innovation is ultimately about.

RE shapes innovative systems:

- On a small scale by striving for exciting new features and ease of use.
- On a large scale by striving for disruptive new ideas.

In 4.2 several techniques for fostering innovation in RE are discussed.

Principle 9 – Systematic and disciplined work: We can't do without in RE

There is a need to employ suitable processes and practices for systematically eliciting, documenting, validating, and managing requirements, regardless of the actual development process being used. Even when a system is developed in an ad-hoc fashion, a systematic and disciplined approach to RE improves the quality of the resulting system.

There is no single process or practice in RE that works well in every given situation or at least in most situations. Systematic and disciplined work means that Requirements Engineers:

- Adapt their processes and practices to the given problem, context, and environment.
- Do not always use the same process and set of practices.
- Do not re-use processes and practices from past successful RE work without reflection.

For every RE endeavor, processes, practices, and work products must be chosen that fit the specific situation best. The details are elaborated in 3, 4, 5 and 6.

3 Work Products and Documentation Practices (L3)

Goal: Understand the fundamental role of work products in RE and be able to create work products

Duration: 7 hours

Terms: Work product, natural-language-based work products, template-based work products, model-based work products, glossary, quality criteria, requirements specification

Educational objectives

EO 3.1.1 Know the characteristics of RE work products and enumerate frequently used types of work products (L1)
EO 3.1.2 Know what each work product can be used for and know the life span of work products (L1)
EO 3.1.3 Explain the different abstraction levels for requirements, including how to choose appropriate abstraction levels and levels of detail (L2)
EO 3.1.4 Know aspects to be considered in work products and the interrelationships between these aspects (L1)
EO 3.1.5 Name the general documentation guidelines (L1)
EO 3.1.6 Describe why it is worth planning the work products to be used (L1)
EO 3.2.1 Know natural-language-based work products and their advantages and disadvantages (L1)
EO 3.2.2 Explain the rules for writing good natural-language-based requirements (L2)
EO 3.3.1 Know the categories of template-based work products and their advantages and disadvantages (L1)
EO 3.3.2 Specify an individual requirement and a user story using a phrase template (L3)
EO 3.3.3 Specify a use case using a form template (L3)
EO 3.4.1 Understand the role, purpose, and use of models in RE (L2)
EO 3.4.2 Understand the advantages and limitations of modeling in RE (L2)
EO 3.4.3 Know the terms model, modeling language, activity model, activity diagram, class model, class diagram, context model, context diagram, domain model, goal model, interaction model, process model, sequence diagram, statechart, state machine, state machine diagram, use case, use case diagram (L1)
EO 3.4.4 Understand how to select an appropriate model type for specifying requirements in a given situation (L2)
EO 3.4.5 Understand and interpret simple models, written in UML where applicable, of the following types: context models, use cases and use case diagrams, domain models, class models, activity models, process models, and statecharts (L2)
EO 3.4.6 Specify a simple model of a system's data or the objects in a domain using a UML class diagram (L3)
EO 3.4.7 Specify a simple system function or business process by a UML activity diagram (L3)
EO 3.5.1 Explain the purpose of glossaries and how to create one (L2)
EO 3.6.1 Know frequently used requirements specification documents (L1)
EO 3.6.2 Explain which document structures serve which purpose and the criteria for structuring (L2)
EO 3.7.1 Know different kinds of prototypes and what they are used for (L1)
EO 3.8.1 Know quality criteria for single requirements (L1)
EO 3.8.2 Know quality criteria for work products (L1)

3.1 Work Products in Requirements Engineering (L2)

A work product is a recorded intermediate or final result generated in a work process. There are a variety of work products in RE, ranging, for example, from short-lived graphic sketches through evolving collections of user stories to formally released contractual requirements specification documents with hundreds of pages.

3.1.1 Characteristics of Work Products (L1)

Work products are characterized by their purpose, representation, size, and life span. The following work products frequently occur in practice for the given purposes. Note that a work product may contain other work products.

- Work products for a single requirement include individual requirements and user stories.
- Work products for a coherent set of requirements include use cases, graphic models of some type (3.4), task descriptions, external interface descriptions, and epics.
- Work products that constitute comprehensive documents or documentation structures include system requirements specifications, product and sprint backlogs, and story maps.
- Other work products include glossaries, textual notes, graphic sketches, and prototypes.

Work products can be *represented* in various forms:

- Natural-language-based (3.2)
- Template-based (3.3)
- Model-based (3.4)
- Other representations, such as drawings or prototypes (3.7)

Most work products are *stored* electronically as files, in databases, or in RE tools. Informal, temporary work products may also be stored on other media — for example, paper or post-it notes on a Kanban board.

When looking at the life span of work products, we distinguish between three categories:

- *Temporary work products:* created to support communication and to create shared understanding.
- *Evolving work products:* typically emerge in several iterations over time; need some metadata (6.5); change control may apply.
- *Durable work products:* have been *baselined* or *released*; need a full set of metadata(6.5); change control must be followed (6.3, 6.4).

A temporary work product may be converted into an evolving work product (by keeping it and adding metadata). Analogously, a temporary or evolving work product can become a durable work product by being baselined or released.

3.1.2 Abstraction Levels (L2)

Requirements usually exist on many *different levels of abstraction* — from, for example, high-level requirements for a new business process down to requirements on a very detailed level, such as the reaction of a specific software component to an exceptional event.

The choice of the proper level of abstraction depends on the subject to be specified and on the purpose of the specification. It is important, however, not to mix requirements that are on different levels of abstraction. In small and medium-sized work products, requirements should be on more or less at the same level of abstraction.

In large work products such as a system requirements specification, requirements on different levels of abstraction should be kept separate by structuring the specification document accordingly (3.6). A requirement at a high level of abstraction may be refined into several detailed requirements at lower, more concrete levels.

When high-level business or stakeholder requirements are expressed in durable work products — such as business requirements specifications, stakeholder requirements specifications, or vision documents — they precede the specification of system requirements. In other settings, business requirements, stakeholder requirements, and system requirements may co-evolve.

3.1.3 Level of Detail (L2)

The level of *detail* to which requirements should be specified depends on several factors, in particular:

- The problem and development context
- The degree of shared understanding of the problem
- The degree of freedom left to designers and programmers
- Availability of rapid stakeholder feedback during design and implementation
- Cost vs. value of a detailed specification
- Imposed standards and regulatory constraints

The greater the level of detail in the requirements specified, the lower the risk of eventually getting something unexpected or unspecified. However, the cost for the specification increases as the level of detail increases.

3.1.4 Aspects to be Considered in Work Products (L1)

When specifying requirements in work products, different aspects need to be considered.

1. Requirements are classified by their kind (1.1) into:
   a) Functional requirements
   b) Quality requirements
   c) Constraints
2. Functional requirements focus on different aspects of the functionality of a system:
   a) Structure and data
   b) Function and flow

c) State and behavior

3. Finally, requirements can only be understood in context (Principle 4 in 2):
   a) S*ystem context*, including external actors
   b) System *boundary* and external interfaces

There are many interrelationships and dependencies between the aspects mentioned. For example, a request issued by a user (context) may trigger a state transition (state and behavior) which initiates an action followed by another action (function and flow) that requires data (structure and data) to provide a result to the user (context) within a given time interval (quality).

Some work products focus on a specific aspect and abstract from the other aspects. This is particularly the case for requirements models (3.4). Other work products, such as a system requirements specification, cover all these aspects. When different aspects are documented in separate work products or in separate chapters of the same work product, these work products or chapters must be kept consistent with each other.

3.1.5 General Documentation Guidelines (L1)

Independently of the techniques used, the following guidelines apply when creating work products:

- Select a work product type that fits the *intended purpose*.
- *Avoid redundancy* by referencing content instead of repeating the same content again.
- *Ensure that there are no inconsistencies* between work products, particularly when they cover different aspects.
- *Use terms consistently*, as defined in the glossary.
- *Structure* work products appropriately.

3.1.6 Plan the Work Products to be Used (L1)

Each project setting and each domain is different, so the set of resulting work products must be defined for each endeavor. Therefore, the following issues must be agreed upon:

- In which work products shall the requirements be recorded and for what purpose?
- Which abstraction levels need to be considered?
- Up to which level of detail must requirements be documented on each abstraction level?
- How shall the requirements be represented in these work products?

The work products to be used should be defined at an early stage in a project. This has several advantages, as it:

- Helps to plan efforts and resources.
- Ensures that appropriate notations are used.

- Ensures that all results are recorded in the right work products.
- Ensures that no major reshuffling of information and "final editing" is needed.
- Helps to avoid redundancy, resulting in less work and easier maintainability.

3.2 Natural-Language-Based Work Products (L2)

Since the inception of systematic RE, natural-language requirements have been a core means for specifying requirements in practice.

Natural-language-based work products have major advantages:

- Unconstrained natural language is extremely expressive and flexible.
- Almost any conceivable requirement in any aspect can be expressed in natural language.
- Natural language is used in everyday life and taught at school, so no specific training is required to read and understand texts in natural language.

These advantages come at the expense that texts written in natural language can frequently be interpreted in different ways, which constitutes a problem when specifying requirements. Furthermore, the detection of ambiguities, omissions, and inconsistencies in such texts is difficult and expensive.

Writing good natural-language-based requirements can be supported by:

- Writing short and well-structured sentences.
- Defining and consistently using a uniform terminology (3.5).
- Avoiding vague or ambiguous terms and phrases.
- Knowing the pitfalls of technical writing listed below.

When writing technical documents in natural language, there are some well-known pitfalls that should be avoided or used with care [GoRu2003].

Things to avoid:

- Incomplete descriptions
- Unspecific nouns
- Incomplete conditions
- Incomplete comparisons

Things to be used with care:

- Passive voice
- Universal quantifiers (such as "all" or "never")
- Nominalizations (i.e., nouns derived from a verb, e.g., "authentication")

3.3 Template-Based Work Products (L3)

Template-based work products are used to overcome some of the shortcomings of natural-language-based work products by providing predefined structures for the requirements.

- *Phrase templates* provide a predefined syntactic structure for a phrase that expresses a requirement, particularly an individual requirement or a user story.
- *Form templates* provide a set of predefined fields in a form to be filled, for example, for writing a use case or a measurable quality requirement.
- *Document templates* provide a predefined structure for a requirements document.

Various templates have been described in literature. [ISO29148], [MWHN2009], and [Rupp2014] provide phrase templates for individual requirements. [Cohn2004] defines a widely used phrase template for user stories and [Cock2001] describes form templates for use cases. [Laue2002] has proposed a template for task descriptions. [ISO29148] and [RoRo2012] provide document templates for whole specifications. Furthermore, a customer might prescribe the use of customer-specific templates in a project.

Advantages of template-based work products:

- Provide a clear, re-useable structure
- Help to capture the most relevant information
- Make requirements and requirements specifications look uniform
- Improve the overall quality of requirements and requirements specifications

Disadvantages and pitfalls of template-based work products:

- People often focus on formal completion of the template rather than on content.
- Aspects that are not included in the template are more likely to be omitted.

3.4 Model-Based Work Products (L3)

Requirements represented in natural language have limitations [Davi1993], in particular with respect to gaining an overview of a set of requirements and understanding the relationships between requirements. Modeling requirements addresses these limitations.

3.4.1 The Role of Models in Requirements Engineering (L2)

A *model* is an abstract representation of an existing part of reality or a part of reality to be created. The notion of reality includes any conceivable set of elements, phenomena, or concepts, including other models. With respect to a model, the modeled part of reality is called the *original*. Examples for models outside the software engineering domain are building information models (BIM) [ISO19650], that model the elements required to plan, build, and manage buildings and other construction elements.

In RE, models help to understand the relationships and interconnections between requirements and provide an overview of a set of requirements. This is primarily achieved by focusing on some aspects — for example, behavior — while abstracting from all other aspects. Gaining an overview is also supported by using a graphic notation for a model. However, models may also be represented in a non-graphical way, for example with tables.

Requirements models have advantages in comparison to requirements represented in natural language:

- Relationships and interconnections between requirements are easier to understand with graphic models than when specifying them in natural language.
- Focusing on a single aspect reduces the cognitive load for understanding the modeled requirements.
- Requirements modeling languages have a restricted syntax which reduces possible ambiguities and omissions.

Models also have limitations:

- Keeping models that focus on different aspects consistent with each other is challenging.
- Information from different models needs to be integrated for causal understanding.
- Models focus primarily on functional requirements; most quality requirements and constraints cannot be expressed in models with reasonable effort.
- The restricted syntax of a graphic modeling language implies that not every relevant item of information can be expressed in a model.

Therefore, requirements models and requirements in natural language are frequently combined.

In RE, models can be used for:

- *Specifying* (primarily functional) requirements in part or even completely, as a means of replacing textually represented requirements.
- *Decomposing* a complex reality into well-defined and complementing aspects; each aspect being represented by a specific model.
- *Paraphrasing* textually represented requirements in order to improve their comprehensibility, in particular with respect to relationships between them.
- *Validating* textually represented requirements with the goal of uncovering omissions, ambiguities, and inconsistencies.

Modeling languages are used to express models. Several modeling languages, for example, UML [OMG2017] or BPMN [OMG2013], have been standardized. When requirements are specified in a non-standard modeling language, a legend is required that explains the syntax and semantics of the modeling language used.

There are many types of models that can be used in RE. A Requirements Engineer must understand which model type is best suited to specify requirements in a given situation.

In an early phase the Requirements Engineer often starts with modeling the context (3.4.2) or the goals of the intended system.

3.4.2 Modeling Context (L2)

Models that focus on the context aspect specify the structural embedding of a system into its environment and the interaction between a system and the actors in the system context.

Context models specify a system and the actors in the system context that interact with the system. A context model also sketches the interfaces between a system and its context (e.g., in terms of what information is exchanged).

Context diagrams are used as a graphic modeling language for expressing context models. There is no standardized notation for context diagrams. Context diagrams from structured analysis [DeMa1978] or tailored box-and-line diagrams [Glin2019] can be used to express context models.

In the modeling language UML [OMG2017], *use case diagrams* provide a means for modeling a system and its context in terms of the system's use cases and the actors in the system context that interact with the system through these use cases.

Use cases model the dynamic interaction between an actor in the system context and a system from the perspective of the actor. Use cases are mostly written using form templates in natural language (3.3) or by using UML activity diagrams (3.4.4).

3.4.3 Modeling Structure and Data (L3)

Models that focus on structure and data aspects specify requirements for static structural properties of a system or a domain.

Static domain models specify the (business) objects and their relationships in a domain of interest. They can be expressed with UML class diagrams [OMG2017].

Class models primarily specify the classes of a system and their attributes and relationships. Classes represent tangible and intangible entities in the real world that the system must know about to fulfill its tasks. UML class diagrams are typically used as a modeling language for class models.

3.4.4 Modeling Function and Flow (L3)

Models that focus on function and flow aspects specify requirements for the sequence of actions required to produce the required results from given inputs or the actions required to execute a (business) process, including the flow of control and data between the actions and who is responsible for which action.

Activity models are used to specify system functions. In the modeling language UML [OMG2017], *activity diagrams* are used to express activity models. They provide elements for modeling actions and the control flow between actions. Activity diagrams can also express who is responsible for which action. Advanced modeling elements (not covered by the CPRE Foundation Level) provide the means for modeling data flow.

Process models are used to describe business processes or technical processes. They can be expressed with UML activity diagrams or with specific process modeling languages such as BPMN [OMG2013]. At the CPRE Foundation Level, we only use UML activity diagrams for process modeling.

3.4.5 Modeling State and Behavior (L2)

Models that focus on state and behavior specify requirements for the behavior of a system or a domain component in terms of state-dependent reactions to events or the dynamics of component interaction.

State machines model events that trigger the transition from one state to another and the actions that need to be performed when a state transition takes place. *Statecharts* [Hare1988] are state machines with states that are decomposed hierarchically and/or orthogonally. State machines, including statecharts, can be expressed in the modeling language UML [OMG2017] with *state machine diagrams* (also called *state diagrams*).

3.4.6 Further Model Types in Requirements Engneering (L1)

At the CPRE foundation level, the understanding and application of models is restricted to selected, important model types. There are further model types that are used in Requirements Engineering. At the CPRE foundation level, it suffices to know them and what they are used for.

Goal models represent a set goals, sub-goals and the relationships between them. Goal models may also include tasks and resources needed to achieve a goal, actors who want to achieve a goal, and obstacles that impede the achievement of a goal.

In SysML [OMG2019], *block definition diagrams* can be adapted to express context diagrams by using stereotyped blocks for the system and the actors. Block definition diagrams can also be used to model the structure of a system in terms of the system's conceptual entities and the relationships between them.

Domain story models can be used to model function and flow, by specifying visual stories about how actors interact with devices, artifacts and other items in a domain, typically using domain-specific symbols [HoSch2020]. They are a means for understanding the application domain in which a system will operate.

Interaction models model the dynamic interactions between objects or actors. UML *sequence diagrams* are a popular means for specifying the interaction between objects.

3.5 Glossaries (L2)

In every RE endeavor that involves more than one person, there is a risk of a lack of a shared understanding of the terminology — that is, that some people interpret the same terms in different ways. To mitigate this problem, the shared understanding of the relevant terms is recorded in a glossary.

A *glossary* is a central collection of definitions for: context-specific terms, everyday terms with a special meaning in the given context, abbreviations, and acronyms. Synonyms (different terms denoting the same thing) should be marked as such. Homonyms (using the same term for different things) should be avoided or marked as such.

The following rules apply for glossaries:

- Manage the glossary centrally.
- Maintain the glossary over the entire course of the system development.
- Define a person or small group who is responsible for the glossary.
- Use a uniform style and structure for the glossary.
- Involve the stakeholders and seek agreement about the terminology.

- Make the glossary accessible for everybody involved.
- Make the use of the glossary mandatory.
- Check work products for proper glossary usage.

3.6 Requirements Documents and Documentation Structures (L2)

Requirements specification documents (3.1.1) comprise several RE work products. It is therefore important to organize such documents with a well-defined structure, in order to create a consistent and maintainable collection of requirements. Besides requirements, a requirements document may also contain additional information and explanations — for example, a glossary, acceptance conditions, project information, or characteristics of the technical implementation.

Requirements may also be organized in documentation structures other than classic documents.

Frequently used documents are:

- Stakeholder Requirements Specification
- User Requirements Specification (a subset of a stakeholder requirements specification, covering only requirements of users)
- System Requirements Specification
- Business Requirements Specification
- Vision Document

Frequently used alternative documentation structures are:

- Product backlog
- Sprint backlog
- Story map

Both the selection of a documentation structure and internal organization of the chosen structure depend on:

- The chosen development process (5)
- The development type and domain
- The contract (a customer may prescribe the use of a given documentation structure)
- The size of the document

Document templates may help to structure a requirements specification. Templates are available in literature [Vole2020], [RoRo2012] and in standards [ISO29148]. Templates may also be re-used from previous, similar projects or may be imposed by a customer. An organization may also decide to create a template as an internal standard.

3.7 Prototypes in Requirements Engineering (L1)

In RE, *prototypes* are a means of specifying requirements by example and of validating requirements. In particular, prototypes can be used if the stakeholders involved do not want

to write and review natural-language-based, template-based or model-based work products.

Exploratory prototypes [LiSZ1994] are used to create shared understanding, clarify requirements, or validate requirements at different levels of fidelity. They are discarded after use.

- *Wireframes* are low-fidelity prototypes built with simple materials or sketching tools that serve primarily for discussing and validating design ideas and user interface concepts.
- *Mock-ups* are medium-fidelity prototypes. When specifying digital systems, they use real screens and click flows, but without real functionality. They serve primarily for specifying and validating user interfaces.
- *Native prototypes* are high-fidelity prototypes that implement critical parts of a system to an extent that stakeholders can use the prototype to see whether the prototyped part of the system will work and behave as expected.

Depending on the degree of fidelity, exploratory prototypes can be an expensive work product, so there is always a trade-off between cost and value gained.

Evolutionary prototypes [LiSZ1994] are pilot systems that form the core of a system to be developed. The final system evolves by incrementally extending and improving the pilot system in several iterations.

3.8 Quality Criteria for Work Products and Requirements (L1)

A requirement needs to meet specific quality criteria to be considered as a good requirement. In modern RE with value-oriented approaches (Principle 1 in 2), the degree of fulfillment for a quality criterion shall correspond to the value created by this requirement. This means that the requirements do not have to adhere to all quality criteria completely — but the higher the value of a requirement, the more relevant are the quality criteria, to reduce the risk of failure.

Adequacy and understandability are the most important quality criteria for single requirements. Without them, a requirement is useless or even harmful, regardless of the fulfillment of all other criteria.

Quality criteria for *single requirements*:

- Adequate (describes true and agreed stakeholder needs)
- Necessary
- Unambiguous
- Complete (self-contained)
- Understandable
- Verifiable

As described in 3.1.1, requirements are usually recorded in various work products that cover single or multiple requirements. The quality criteria above shall be used to create high-

quality single requirements in a work product. For work products that cover more than one requirement, the following quality criteria shall be considered additionally.

Quality criteria for work products covering *multiple requirements*:

- Consistent
- Non-redundant
- Complete (no known and relevant requirements missed)
- Modifiable
- Traceable
- Conformant

4 Practices for Requirements Elaboration (L3)

Goal: Understand the use of practices to identify requirements sources, to elicit requirements, to identify and resolve conflicts, and to validate requirements

Duration: 4 hours 30 minutes

Terms: Requirements source, system boundary, system context, requirements elicitation, requirements validation, stakeholder, Kano model, conflict resolution

Educational objectives

EO 4.1.1 Determine the boundaries of the system to focus on the relevant requirements (L3)
EO 4.1.2 Remember the relevant sources for the system to be created (L1)
EO 4.1.3 Identify stakeholders and write a stakeholder list (L3)
EO 4.1.4 Understand the benefits of stakeholder management (L2)
EO 4.2.1 Understand how the Kano model can help to elicit the right requirements (L2)
EO 4.2.2 Understand the difference between gathering techniques, and design and idea-generating techniques (L2)
EO 4.2.3 Understand how to choose a proper elicitation technique for a given situation (L2)
EO 4.3.1 Remember the different types of conflict (L1)
EO 4.3.2 Understand what activities are necessary to solve conflicts (L2)
EO 4.3.3 Understand how to apply appropriate conflict resolution techniques (L2)
EO 4.4.1 Understand why requirements documents need to be validated (L2)
EO 4.4.2 Remember the four important aspects for requirements validation (L1)
EO 4.4.3 Understand how to apply appropriate techniques for requirements validation (L2)

4.1 Sources for Requirements (L3)

The quality and completeness of requirements depend greatly on the requirements sources involved. Missing a relevant source will lead to an incomplete understanding of the requirements or to incomplete requirements. Identification of requirements sources is an iterative and recursive process that requires constant reconsideration.

A shared understanding (Principle 3 in 2) of the context of the system to be developed is a prerequisite for being able to identify the relevant requirements sources. The area between the system boundary and the context boundary is called the (system) context (Principle 4 in 2). The (system) context is needed to understand the nature of the requirements to be developed and thus to identify the original sources for requirements.

Requirements sources are classified in three types:

- Stakeholders
- Documents
- Systems

The stakeholders of a system (see [Glin2020] for a definition; see also Principle 2 in 2) are the main source for requirements. Typical stakeholder roles include [BiSp2003]:

- Users (also called end-users)
- Sponsors
- Managers
- Developers
- Authorities
- Customers

Furthermore, people or organizations that are *impacted* by a system should be considered as (indirect) stakeholders.

Systematic identification of stakeholders should take place at the beginning of a development venture and the results should be managed throughout development as a continuous activity. This includes the identification of both stakeholder roles and persons in these roles.

For all systems with a user interface, the *end users* of the system constitute a stakeholder group which is of particular interest for the requirements engineer. End users should be aggregated into groups (for example, by similar roles, tasks or responsibilities).

When end users can be identified individually, representatives from each groups should be chosen. Otherwise, personas can be defined for representing the relevant end user groups [Coop2004].

Potential sources for identifying relevant stakeholders and stakeholder roles are:

- Checklists of typical stakeholder groups and roles
- Organizational structures
- Business process documentation
- Market reports
- Initial stakeholders for identifying *additional* stakeholders

Stakeholders should be documented in an up-to-date stakeholder list with (at least) the following information:

- Name
- Function (role)
- Additional personal and contact data
- Temporal and spatial availability during the project progress
- Relevance
- Area and extent of expertise
- Goals and interests in terms of the project

Problems with stakeholders may arise if the rights and obligations of a stakeholder are not clear or if the stakeholder's needs are not sufficiently addressed. Stakeholder relationship management [Bour2009] is an effective way to counter problems with stakeholders.

In most system contexts, more sources are available. They must also be considered for a successful new system, as most stakeholders do not talk about the obvious: their "subconscious" requirements (4.2).

Additional sources for requirements include:

- Existing and legacy systems
- Process documents
- Legal or regulatory documents
- Company-specific regulations
- (Marketing) information about potential future users

Another source for requirements can be found by looking at similar situations in completely different domains.

4.2 Elicitation of Requirements (L2)

Within elicitation, it is the task of the Requirements Engineer to understand the stakeholders' desires and needs while ensuring that the requirements from all relevant requirements sources have been collected, by applying appropriate techniques to elicit them. A major point in elicitation is turning implicit demands, wishes, and expectations into explicit requirements.

To elicit requirements, it is crucial to know the nature and importance of each requirement. These may change from project to project and also over time. The Kano model [KaeA1984] classifies requirements into three relevant categories:

- Delighters (synonyms: excitement factors, unconscious requirements)
- Satisfiers (synonyms: performance factors, conscious requirements)
- Dissatisfiers (synonyms: basic factors, subconscious requirements)

There are many different techniques for eliciting these categories of requirements. We differentiate between:

- Gathering techniques
- Design and idea-generating techniques

Gathering techniques are established techniques for requirements elicitation [BaCC2015] that help to elicit satisfiers and dissatisfiers by investigating different sources.

Four main categories can be discerned:

- Questioning techniques
- Collaboration techniques
- Observation techniques
- Artifact-based techniques

Design and idea-generating techniques are intended to stimulate creativity during requirements elicitation. They aim at creating ideas for solving a problem and at exploring design ideas [Kuma2013]. This may lead to new or innovative requirements that are often delighters. Popular examples of such techniques are brainstorming [Osbo1979], analogies, prototyping (e.g., mock-ups), scenarios, and storyboards.

A broader concept related to design and idea generation is *design thinking*. Different approaches exist, such as *d.school* [Sdsc2012] and *Designing for Growth* [LiOg2011], offering a wide set of techniques that can be used for elicitation of requirements.

Elicitation techniques should be able to detect all kinds of requirements — functional and quality requirements, and constraints alike. In practice, quality requirements and constraints often get less attention.

To elicit *quality requirements*, a quality model such as the ISO 25010 standard [ISO25010] should be used as a checklist. This model can also be helpful in quantifying requirements.

Constraints can be found by considering possible restrictions of the potential solution space — for example, technical, legal, organizational, cultural, or environmental issues.

Picking the right elicitation techniques is a critical key competence that depends on many different factors, such as:

- Type of system
- Software development life cycle model
- People involved
- Organizational setup

The best results are usually achieved with a combination of different elicitation techniques. [CaDJ2014] represent a systematic approach for selecting the techniques.

4.3 Resolving Conflicts regarding Requirements (L2)

Elicitation techniques by themselves do not ensure that the resulting set of requirements as a whole is consistent, complete, conformant, etc. (3.8). For the final set, however, all stakeholders must understand and agree on all requirements that are relevant to them. If some stakeholders do not agree, this situation is to be recognized as a conflict that should be resolved accordingly. Suitable conflict resolution techniques should be selected based on the type of conflict and contextual information. This requires an in-depth understanding of the nature of the requirements conflict and of the attitudes of the stakeholders involved.

Tasks to identify and resolve conflicts are:

- Conflict identification
- Conflict analysis
- Conflict resolution
- Documentation of conflict resolution (decisions made)

It is useful to distinguish between different conflict types [Moor2014]. The following types of conflict often require attention from the Requirements Engineer:

- Subject matter conflict
- Data conflict
- Interest conflict
- Value conflict
- Relationship conflict

- Structural conflict

To resolve conflicts successfully, common techniques can be applied:

- Agreement
- Compromise
- Voting
- Overruling
- Definition of variants

In addition, there are several auxiliary techniques, for example:

- Consider-all-facts
- Plus-minus-interesting
- Decision matrix

4.4 Validation of Requirements (L2)

Validating requirements is an important step towards a successful system (Principle 6 in 2). Ensuring requirements quality up front will reduce wasted effort downstream. Validating requirements means checking the work products at hand as well as the individual requirements in it for quality (see 3.8 for details).

Important aspects to be considered in requirements validation are:

- Involvement of the right stakeholders
- Separating the identification and the correction of defects
- Validation from different views
- Repeated validation

There are several techniques for validation (e.g. [GiGr1993], [OleA2018]). These validation techniques are often classified into:

- *Review techniques*, including:
  - Walkthroughs
  - Inspections
- *Exploratory techniques*, for example:
  - Prototyping
  - Alpha testing and beta testing
  - A/B testing [KoTh2017]
  - Building a Minimum Viable Product (MVP)
- *Sample development*

These techniques differ in formality and effort. Which technique to select depends on factors such as the software development life cycle model, the maturity of the development process, the complexity and risk level of the system, any legal or regulatory requirements, and/or the need for an audit trail.

5 Process and Working Structure (L3)

Goal: Explain the concepts of an RE process and apply appropriate process configurations

Duration: 1 hour 15 minutes

Terms: Process, RE process

Educational objectives

EO 5.1.1 Know the important factors that influence an RE process (L1)
EO 5.1.2 Understand how and why these factors are influencing (L2)
EO 5.2.1 Understand the facets to be considered for the configuration of an RE process (L2)
EO 5.3.1 Know typical RE process configurations (L1)
EO 5.3.2 Understand the steps for configuring an RE process (L2)
EO 5.3.3 For simple system and development settings, select and apply an appropriate RE process configuration (L3)

A process is required to shape and structure the RE work to be done in a given context. As there is no one-size-fits-all RE process (1.4), a tailored RE process must be configured that fits the given development and system context.

The RE process shapes the information flow and the communication model between various participants (e.g., customers, users, Requirements Engineers, developers, testers) and also defines the work products to be used or produced. Thus, the RE process provides the framework for eliciting, documenting, validating, and managing requirements.

5.1 Influencing Factors (L2)

Many factors influence the configuration of an RE process. The main factors are:

- Overall process fit: the RE process must fit the overall system development process.
- Development context
- Capability and availability of stakeholders
- Shared understanding
- Complexity and criticality of the system to be developed
- Constraints
- Available time and budget
- Volatility of requirements
- Experience of Requirements Engineers

An analysis of the influencing factors provides information about how to configure the RE process. The influencing factors also constrain the space of possible process configurations. For example, when the stakeholders are only available at the beginning of the project, no process can be chosen that builds upon continuous stakeholder feedback.

5.2 Requirements Engineering Process Facets (L2)

There are three decisive facets that need to be considered when configuring an RE process [Glin2019].

Time Facet: Linear versus Iterative

In a linear process, requirements are specified up front in a single phase of the process. In an iterative process, requirements are specified incrementally, starting with general goals and some initial requirements, and then adding or modifying requirements in every iteration.

Criteria for choosing a *linear* RE process:

- The development process for the system is plan-driven and mostly linear.
- The stakeholders know their requirements and can specify them up front.
- A comprehensive requirements specification is required as a contractual basis for outsourcing the design and implementation of the system.
- Regulatory authorities require a comprehensive, formally released requirements specification at an early stage of the development.

Criteria for choosing an *iterative* RE process:

- The development process for the system is iterative and agile.
- Many requirements are not known up front, but will emerge and evolve during the development of the system.
- Stakeholders are available such that short feedback loops can be established as a means of mitigating the risk of developing the wrong system.
- The duration of the development allows for more than just one or two iterations.
- The ability to change requirements easily is important.

Purpose Facet: Prescriptive versus Explorative

In a prescriptive RE process, the requirements specification constitutes a contract: all requirements are binding and must be implemented. In an explorative RE process, only the goals are known a priori, while the concrete requirements have to be explored.

Criteria for choosing a *prescriptive* RE process:

- The customer requires a fixed contract for system development.
- Functionality and scope take precedence over cost and deadlines.
- The development of the specified system may be tendered or outsourced.

Criteria for choosing an *explorative* RE process:

- Stakeholders initially only have a vague idea about their requirements.
- Stakeholders are strongly involved and provide continuous feedback.
- Deadlines and cost take precedence over functionality and scope.
- It is not a priori clear which requirements actually shall be implemented and in which order they will be implemented.

Target Facet: Customer-Specific versus Market-Oriented

In a customer-specific RE process, the system is ordered by a customer and developed by a supplier. In a market-oriented RE process, the system is developed as a product or service for a market, targeting specific user segments.

Criteria for choosing a *customer-specific* RE process:

- The system will be mainly used by the organization that has ordered the system and pays for its development.
- The important stakeholders are mainly associated with the customer's organization.
- Individual persons can be identified for the stakeholder roles.
- The customer wants a requirements specification that can serve as a contract.

Criteria for choosing a *market-oriented* RE process:

- The developing organization intends to sell the system in some market segment as a product or service.
- Prospective users are not individually identifiable.
- The Requirements Engineers have to design the requirements so that they match the envisaged needs of the targeted users.
- Product owners, marketing people, digital designers, and system architects are the primary stakeholders.

Hints and Caveats

- The criteria presented above are *heuristics* rather than fixed rules. For example, outsourcing the development of the system is done preferably with a prescriptive RE process rather than with an explorative one because the contract between the customer and the supplier is typically based on a comprehensive requirements specification. However, it is also possible to negotiate an outsourcing contract based on an explorative RE process.
- Linear RE processes work only if a sophisticated process for changing requirements is in place.
- Linear RE processes imply long feedback loops: To mitigate the risk of developing the wrong system, requirements must be validated intensively.
- When defining an RE process, *linear* and *prescriptive* are frequently chosen together.
- Explorative RE processes are typically also iterative processes (and vice versa).
- In a market-oriented process, feedback from users is the only means of validating whether the product will actually satisfy the needs of the targeted user segment.
- The market-oriented facet does not combine well with the linear and prescriptive facets.

5.3 Configuring a Requirements Engineering Process (L3)

In a concrete system development context, the persons responsible for RE have to configure the RE process to be applied. Based on an analysis of the influencing factors in 5.1,

a suitable combination of the process facets described in 5.2 can be used [Glin2019]. Below, three typical combinations are described.

Participatory RE Process: Iterative & Exploratory & Customer-Specific

Main application case:	Supplier and customer collaborate closely; stakeholders are strongly involved both in the RE and the development processes
Typical work products:	Product backlog with user stories and/or task descriptions, prototypes
Typical information flow:	Continuous interaction between stakeholders, product owners, Requirements Engineers, and developers; may include feedback from users

Contractual RE Process: Typically Linear (Sometimes Iterative) & Prescriptive & Customer-Specific

Main application case:	The requirements specification constitutes the contractual basis for the development of a system by people not involved in the specification and with little stakeholder interaction after the requirements phase.
Typical work products:	Classic system requirements specification, consisting of natural-language-based requirements and models
Typical information flow:	Primarily from stakeholders to Requirements Engineers

Product-Oriented RE Process: Iterative & Explorative & Market-Oriented

Main application case:	An organization specifies and develops software in order to sell or distribute it as a product or service
Typical work products:	Product backlog, prototypes
Typical information flow:	Interaction between product owner, marketing, Requirements Engineers, digital designers, developers and (maybe) fast feedback from customers/users

Note that there may be system and development contexts where none of the aforementioned configurations fit. For example, regulatory constraints may impose the use of a process that conforms to given standards such as ISO/IEC/IEEE 29148 [ISO29148].

When configuring an RE process, we recommend to use a five-step procedure:

1. Analyze the influencing factors (5.1)
2. Assess the facet criteria (5.2)
3. Configure the process (5.3)
4. Determine work products (3)
5. Select appropriate practices

6 Management Practices for Requirements (L2)

Goal: Understand the need for and benefit of requirements management

Duration: 2 hours

Terms: Requirements management, change management, traceability, requirements attributes, requirements life cycle, prioritization

Educational objectives

EO 6.1.1 Know what requirements management is about and why is it needed (L1)
EO 6.2.1 Explain why requirements work products need a status/life cycle model (L2)
EO 6.3.1 Explain what a requirements versioning concept looks like in a given project situation (L2)
EO 6.4.1 Know the use of requirements configurations and baselines (L1)
EO 6.5.1 Know the purpose of attributes for requirements (L1)
EO 6.5.2 Explain what an appropriate set of attributes for requirements looks like in a given project situation (L2)
EO 6.5.3 Explain the purpose of views and name the different views of requirements (L2)
EO 6.6.1 Name reasons for requirements traceability (L1)
EO 6.6.2 Summarize the differences between implicit and explicit traceability (L1)
EO 6.6.3 Know how explicit traceability can be documented (L1)
EO 6.7.1 Know how to handle changes in linear (plan-driven) and agile approaches (L1)
EO 6.8.1 Know the reason for prioritization and know meaningful assessment criteria (L1)
EO 6.8.2 Name steps to prioritize requirements (L1)
EO 6.8.3 Name different categories of prioritization techniques (L1)

6.1 What is Requirements Management? (L1)

Requirements management is the process of managing existing requirements recorded in various work products. In particular, this includes storing, changing, and tracing requirements [Glin2020]. Requirements management can happen in different ways and at different levels depending on the chosen development process and context — see, for example, [Leff2011], [Rupp2014], [WiBe2013]. Regardless of the circumstances, the task of requirements management is to maintain requirements in such a way, that all roles in a project can work effectively and efficiently.

6.2 Life Cycle Management (L2)

Life cycle management refers to the process of keeping track of all work products with respect to status in their life cycle. Each documented requirement and each work product has its own life cycle: it is created, then evaluated and refined before it is reviewed, reworked, consolidated, agreed, and so forth. To allow identification of which work product is in which state, a life cycle model is required defining each allowed life cycle status and state transition. The actual status of a work product should always be clear, including (usually) the history of its transitions.

6.3 Version Control (L2)

Version control of requirements refers to the process of keeping track of all work products during their evolution. Any change in a work product should be reflected by a new version. Versioning allows the history of a work product to be traced back to its origin and the restoring of a work product to any earlier version. For this purpose, version control requires three measures to be in place:

- A version number to uniquely identify the version of a work product.
- A history of what was changed.
- A concept for work product storage.

Versioning must be considered for all work products [WiBe2013]. A version number typically consists of at least two parts: the version and the increment.

6.4 Configurations and Baselines (L1)

A requirements *configuration* is a consistent set of work products that contain requirements. Each configuration is defined for a specific purpose and contains at most one version of each work product [Glin2020]. The purpose of configurations is, for example, to review a set of work products or to facilitate an estimation of the development effort.

A *baseline* is a stable, change-controlled *configuration* of work products, used for release planning or other delivery milestones in a project [Glin2020].

Configurations have the following properties:

1. Logical connection
2. Consistency
3. Uniqueness
4. Unchangeability
5. Basis for reset

6.5 Attributes and Views (L2)

Attributes are required to document important metadata for a work product and are typically used to answer a number of important questions during the project or product life cycle.

The objective of using attributes to characterize requirements is to enable team members and other stakeholders to get the information about the requirements they need at any point during the project.

Defining the relevant set of attributes depends on the information needs of the different stakeholders in the project. Existing standards, for example [ISO29148], provide an overview of the most relevant attributes.

Views are an excerpt from the total set of requirements that contain only the content that is currently of interest. From a technical perspective, a view is a combination of filter and sort

settings that can be made available or reusable to other participants by storing the selected combination.

We differentiate between three types of views:

- *Selective views*
- *Projective views*
- *Aggregating views*

In most cases, views of requirements are combinations of selective, projective, and aggregating views for the creation of reports.

6.6 Traceability (L1)

Traceability [GoFi1994] is the ability to trace a requirement *back to its origin* (i.e., stakeholders, documents, justifications, etc.) and *forward to subsequent work products (e.g., test cases)*, as well as *to other requirements* that this requirement depends on.

Traceability is a prerequisite for requirements management and is often explicitly demanded by standards, laws and regulations. Implementing traceability basically means maintaining dependencies between different work products (3.1) at different abstraction levels (3.1.2), levels of detail (3.1.3) and to all relevant predecessors and successors for reasons of analysis, compliance and information.

Traceability can be documented *implicitly,* by structuring and standardizing work products*,* or *explicitly,* by relating work products to each other based on their unique identifiers in various forms [HuJD2011]. Common representation forms are hyperlinks, references, matrices, tables, or graphs.

6.7 Handling Change (L1)

Requirements are not static. Changes in requirements happen due to many different reasons and need to be handled properly (Principle 7 in 2) for instance by creating a formal *change request*, or by adding a new item to the *product backlog*.

Decision making, planning and controlling implementation of a change depends on the development approach and the point in time the change occurs.

In a *linear* approach, the decision on a change is often taken by a Change Control Board (in projects) or a Change Advisory Board (in operation). In a more *iterative* approach, the product owner includes the change in the product backlog and prioritizes the new item accordingly.

6.8 Prioritization (L1)

Not all requirements are equally important [Davi2005]. Assessment and prioritization are used to determine the most relevant requirements for the next product release or increment.

The *assessment* of the requirements is the basis for their prioritization, often determined by using multiple assessment criteria such as business value, urgency, effort, dependencies and others.

The *priority* of a requirement describes the importance of a single requirement compared to other requirements according to certain criteria [Glin2020]. The *prioritization* itself is performed based on a single criterion or on multiple criteria; this depends mainly on the prioritization technique chosen.

Steps for prioritization:

- Define major goals and constraints for the prioritization
- Define desired assessment criteria
- Define the stakeholders that have to be involved
- Define the requirements that have to be prioritized
- Select the prioritization technique
- Perform prioritization

Prioritization techniques can be classified in:

- *Ad-hoc* prioritization techniques
- *Analytical* prioritization techniques

7 Tool Support (L2)

Goal: Provide an overview of the role of RE tools and aspects for the implementation

Duration: 30 minutes

Terms: Tool, RE tool

Educational objectives

EO 7.1.1 Know the different types of RE tools (L1)
EO 7.2.1 Explain what to consider when introducing RE tools (L2)

7.1 Tools in Requirements Engineering (L1)

The RE process can be supported by tools that support dedicated tasks and activities. As the RE process is quite individual (5), existing RE tools often focus only on certain aspects in RE and rarely support all activities. Before selecting a tool, Requirements Engineers should decide which tasks and activities during the RE process should be supported and how. We differentiate between tooling that supports:

- Management of requirements:
  - Defining and storing requirements attributes
  - Prioritizing requirements
  - Managing versions and configurations
  - Tracking and tracing requirements
  - Managing changes to requirements
- Management of the RE process:
  - Measuring and reporting on the RE process
  - Measuring and reporting on the product quality
  - Managing the RE workflow
- Documentation of knowledge about the requirements:
  - Sharing requirements
  - Creating a shared understanding of the requirements
- Modeling of requirements
- Collaboration in RE
- Testing/simulation of requirements

Tools often come in a mixture of the above-mentioned features. To ensure that all RE tasks are covered appropriately, different tools might be combined.

Sometimes, other kinds of tools, for example office or issue tracking tools, are used to document or manage requirements. These tools have limitations and should only be used when the RE process is in control and requirements are aligned and quite stable.

7.2 Introducing Tools (L2)

Selecting an RE tool is no different from selecting a tool for any other purpose. The goal, context, and requirements must be described before selection can be successful [Fugg1993].

An appropriate tool can only be sought once the proper RE procedures and techniques have been introduced. Tool introduction requires clear RE responsibilities and procedures. In the process of introducing an RE tool, the following aspects are relevant:

- Take into account all life cycle costs beyond license costs
- Consider necessary resources
- Use pilot projects to avoid risks
- Evaluate the tool according to defined criteria
- Instruct employees in how to use the tool

References

[AnPC1994] Annie I. Antón, W. Michael McCracken, Colin Potts: Goal Decomposition and Scenario Analysis in Business Process Reengineering. CAiSE (Conference on Advanced Information Systems Engineering), 1994, 94–104.

[BaCC2015] K. Baxter, C. Courage, K. Caine: Understanding Your Users: A Practical Guide to User Research Methods, 2nd edition. Morgan Kaufmann, Burlington, 2015.

[BiSp2003] K. Bittner, I. Spence: Use Case Modelling. Pearson Education, Boston, 2003.

[Bour2009] L. Bourne: Stakeholder Relationship Management: A Maturity Model for Organisational Implementation. Gower Publishing Ltd, Birlington, 2009.

[CaDJ2014] D. Carrizo, O. Dieste, N. Juristo: Systematizing requirements elicitation technique selection. Information and Software Technology 2014, 56(6): 644–669.

[Cock2001] A. Cockburn: Writing Effective Use Cases. Addison-Wesley, Boston 2001.

[Cohn2004] M. Cohn: User Stories Applied – For Agile Software Development. Addison-Wesley, Boston, 2004.

[Coop2004] A. Cooper: The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity. Que, Indianapolis, 2004.

[Davi2005] A. M. Davis: Just Enough Requirements Management – Where Software Development Meets Marketing. Dorset House Publishing, New York, 2005.

[Davi1993] A. M. Davis: Software Requirements – Objects, Functions, & States, 2nd edition, Prentice Hall, New Jersey, 1993.

[DeMa1978] T. DeMarco: Structured Analysis and System Specification. Yourdon Press, New York, 1978.

[Fugg1993] A. Fuggetta: A classification of CASE technology. IEEE Computer 1993, 26 (12): 25–38.

[GiGr1993] T. Gilb, D. Graham: Software Inspection. Addison Wesley, Boston, 1993.

[Glin2019] M. Glinz: Requirements Engineering I. Course Notes, University of Zurich, 2019. https://www.ifi.uzh.ch/en/rerg/courses/archives/hs19/re-i.html#resources. Last visited July 2020.

[Glin2020] M. Glinz: A Glossary of Requirements Engineering Terminology. Version 2.0. https://www.ireb.org/en/downloads/#cpre-glossary. Last visited July 2020.

[GoFi1994] O. Gotel, A. Finkelstein: An Analysis of the Requirements Traceability Problem. 1st International Conference on Requirements Engineering, Colorado Springs, 1994. 94–101.

[GoRu2003] R. Goetz, C. Rupp: Psychotherapy for System Requirements. 2nd IEEE International Conference on Cognitive Informatics (ICCI'03), London, 2003. 75–80.

[GRL2020] Goal oriented Requirement Language. University of Toronto, Canada https://www.cs.toronto.edu/km/GRL. Last visited May 2020.

[Hare1988] D. Harel: On Visual Formalisms. Communications of the ACM 1988, 31 (5): 514–530.

[HoSch2020] S. Hofer, H. Schwentner: Domain Storytelling — A Collaborative Modeling Method. Available from Leanpub, http://leanpub.com/domainstorytelling. Last visited July 2020.

[HuJD2011] E. Hull, K. Jackson, and J. Dick: Requirements Engineering. Springer, 3rd Ed, 2011.

[ISO29148] ISO/IEC/IEEE 29148: Systems and Software Engineering – Life Cycle Processes – Requirements Engineering, International Organization for Standardization, 2018.

[ISO19650] ISO 19650: Organization and Digitization of Information about Buildings and Civil Engineering Works, including Building Information Modelling (BIM)– Information Management Using Building Information Modelling – Part 1 and 2, International Organization for Standardization, 2018.

[ISO25010] ISO/IEC/IEEE25010:2011: Systems and software Quality Requirements and Evaluation (SQuaRE) –– System and software quality models. International Organization for Standardization, Geneva, 2011.

[Jack1995] M. A. Jackson: Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices. Addison-Wesley, New York, 1995.

[Jack1995b] M. Jackson: The World and the Machine. 17th International Conference on Software Engineering 1995 (ICSE 1995). 287–292.

[KaeA1984] N. Kano et al.: Attractive quality and must-be quality. Journal of the Japanese Society for Quality Control 1984, 14(2): 39–48. (in Japanese)

[KoTh2017] R. Kohavi, S. Thomke: The Surprising Power of Online Experiments – Getting the most out of A/B and other controlled tests. Harvard Business Review, Sept–Oct 2017: 74–82.

[Kuma2013] V. Kumar: 101 Design Methods – A Structured Approach for Driving Innovation in Your Organization. John Wiley & Sons, Hoboken, 2013.

[Laue2002] S. Lauesen: Software Requirements. Styles and Techniques. Addison-Wesley, Harlow, 2002.

[Leff2011] D. Leffingwell: Agile Software Requirements, Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley, Boston, 2011.

[LiOg2011] J. Liedtka, T. Ogilvie: Designing for Growth: A Design Thinking Tool Kit For Managers. Columbia University Press, 2011.

[LiSZ1994] H. Lichter, M. Schneider-Hufschmidt, H. Zullighoven: Prototyping in Industrial Software Projects – Bridging the Gap Between Theory and Practice. IEEE Transactions on Software Engineering 1994, 20 (11): 825–832.

[MFeA2019] D. Méndez Fernández, X. Franch, N. Seyff, M. Felderer, M. Glinz, M. Kalinowski, A. Volgelsang, S. Wagner, S. Bühne, K. Lauenroth: Do We Preach What We Practice? Investigating the Practical Relevance of Requirements Engineering Syllabi – The IREB Case. CIbSE 2019: 476–487.

[Moor2014] C. W. Moore: The Mediation Process – Practical Strategies for Resolving Conflicts, 4th edition. John Wiley & Sons, Hoboken, 2014.

[MWHN2009] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak: Easy Approach to Requirements Syntax (EARS). 17th IEEE International Requirements Engineering Conference (RE'09), Atlanta, Georgia, 2009. 317–322.

[OleA2018] K. Olsen et al.: Certified Tester, Foundation Level Syllabus – Version 2018. International Software Testing Qualifications Board, 2018.

[OMG2013] Object Management Group: Business Process Model and Notation (BPMN), version 2.0.2. OMG document formal/2013-12-09 http://www.omg.org/spec/BPMN. Last visited July 2020.

[OMG2017] Object Management Group: OMG Unified Modeling Language (OMG UML), version 2.5.1. OMG document formal/2017-12-05. https://www.omg.org/spec/UML/About-UML/. Last visited July 2020.

[OMG2019] Object Management Group: OMG Systems Modeling Language (OMG SysML™), Version 1.6. OMG Document formal/19-11-01. https://www.omg.org/spec/SysML/. Last visited January 2022.

[Osbo1979] A. F. Osborn: Applied Imagination, 3rd revised edition. Charles Scribner´s Sons, New York, 1979.

[RoRo2012] S. Robertson and J. Robertson: Mastering the Requirements Process, 3rd edition. Addison-Wesley, Boston, 2012.

[Rupp2014] C. Rupp: Requirements-Engineering und Management, 6. Auflage. Hanser, München, 2014. (in German).

[Sdsc2012] Stanford d.school: An Introduction to Design Thinking. Hasso Plattner Institute of Design, Stanford, 2012. https://dschool-old.stanford.edu/groups/designresources/wiki/36873. Last visited July 2020.

[vLam2009] Axel van Lamsweerde: Requirements Engineering: From System Goals to UML Models to Software Specifications. Chichester: John Wiley & Sons, 2009.

[Vole2020] Volere: Requirements Resources. https://www.volere.org. Last visited July 2020.

[WiBe2013] K. Wiegers and J. Beatty: Software Requirements, 3rd edition. Microsoft Press, Redmond, 2013.