IREB Certified Professional for Requirements Engineering

- Requirements Management, Advanced Level -

Syllabus

Version 1.1.0

September 11, 2019

Acknowledgements

This syllbus was created by (in alphabetical order) Stan Bühne, Frank Engel, Sven Eselgrimm, Günter Halmans, Andrea Herrmann, Frank Houdek, Patrick Mäder, Alexander Rachmann, Thomas Schölzl, Amin Soesanto, Frank Stöckel and Malik Tayeh.

We thank everybody for their honorary involvement.

Preamble

Purpose of the document

This syllabus defines the advanced level of the Requirements Management module 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 Advanced Requirements Management module is aimed at professionals in the fields of requirements engineering, business analysis, business engineering, organizational design, etc. who wish to deepen their knowledge and skills in the area of requirements management.

Content Scope

The advanced level - as does the foundation level - also teaches basic principles that are equally valid for all domains (e.g. embedded systems, safety-critical systems, classical information systems). This does not mean that the suitability of approaches for the individual areas cannot be handled in training courses that consider the special characteristics of those areas. It is not, however, the goal to present requirements engineering specific to a certain domain.

The contents covered in this syllabus can be used in the same way in (development) projects as in product management; in the evolution of existing systems or in the continuous, cross-project management of requirements. For reasons of simplicity, often only projects are mentioned instead of this extensive list.

No particular set of procedures or associated process model is assumed, that would prescribe the planning, steering and sequence of application of the learned concepts in practice.

It is not the aim to promote a particular process for requirements engineering, or for software or systems engineering in general. However, in EU 10 specific aspects of Requirements Management in agile projects are discussed.

Requirements management without the use of tools is difficult in practice. Therefore, the possibilities and limitations of tool support are discussed in the individual chapters without, however, placing a special tool in the foreground.

Level of Detail

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

- General educational objectives
- Contents with a description of the learning objectives and, where necessary, references to further literature.

Educational Objectives / Cognitive Knowledge Levels

Each module of the syllabus is assigned a cognitive level. A higher level includes the lower levels. In the formulations of the educational objectives the verbs "knowing", for level K1, and "mastering and using", for level K2, are representative of the following verbs in the respective lists below.

- **L1 (knowing)**: enumerate, characterize, recognize, name, reflect
- **L2 (mastering and using):** analyze, use, execute, justify, describe, judge, display, design, develop, complete, explain, exemplify, elicit, formulate, identify, interpret, conclude from, assign, differentiate, compare, understand, suggest, summarize

All terms defined in the glossary have to be known (L1), even if they are not explicitly mentioned in the educational objectives.

This syllabus uses the abbreviation "RE" for Requirements Engineering.

Structure of the Syllabus

The syllabus consists of 11 main chapters. One chapter covers one educational unit (EU). Each main chapter title contains the cognitive level of the chapter, which is the highest level of the sub-chapters. Furthermore, the minimum teaching time a course should invest for that chapter is suggested. Important terms in the chapter, which are defined in the glossary, are listed at the beginning of the chapter.

Example: EU 3 Assigning Attributes and Views for Requirements (K2)
Duration: 2 hours
Terms: Attribute, Attribute Schema, View

Educational Objectives:

EO 3.1 Knowing the objective of assigning attributes and the benefits of assigning attributes in management activities (K1)
EO 3.2 Knowing advantages of attribute schemas (K1)
EO 3.3.1 Knowing the steps for designing an attribute schema (K1)
EO 3.3.2 Mastering and using predefined steps to design an attribute schema (K2)
…

EO 3.7 Evaluating the degree to which an attribute is populated (K2)

The example shows that the teaching unit Assigning Attributes and Views for Requirements with a duration of 2 hours is dealt with in chapter 3. The terms attribute, attribute schema and view or their definitions from the glossary must be known.

The first sub-chapter, unit 3.1, is about knowing the objective of assigning attributes and also the benefits of assigning attributes in management activities, and contains the first learning objective LO 3.1.1 with level K1. It is therefore important to be able to describe the goals of assigning attributes as well as the use cases in management activities. The learning objective LO 3.3.2, for example, has level K2, so it is not only a question of knowing the content, but also of being able to use it.

As the Advanced Level is a deepening and extension of the Foundation Level, there is also a need in the syllabus to briefly repeat the contents of the Foundation Level in order to create a basis for extension and deepening. These overlapping contents are shown in the syllabus with a blue bar on the right-hand side of the page. These contents are also relevant for the examination.

The Examination

This syllabus is the basis for the examination for the Requirements Management, Advanced Level certificate.

A question in the examination can cover material from several chapters of the syllabus. All chapters of the syllabus can be examined.

The examination consists of multiple choice exercises as well as an evaluated homework. The examination rules regulate the details.

Examinations can be held immediately after a training course, but also independently from courses (e.g. in an examination center). The certification bodies licensed by the IREB are listed on the website: www.ireb.org

Content

EU 1 What is Requirements Management? (K1)

Duration: 1 ¼ hours
Terms: Requirements Engineering, Requirements Management, Requirements Manager, Requirements Management Plan

Educational Objectives:
EO 1.1 Knowing the definition of Requirements Engineering and Requirements Management (K1)
EO 1.2 Knowing Requirements Management tasks (K1)
EO 1.3 Knowing the goals and benefits of Requirements Management (K1)
EO 1.4 Knowing the necessity of a Requirements Management Plan (K1)
EO 1.5 Knowing the relevant standards for Requirements Management (K1)

EU 1.1 Definition of Requirements Managements (K1)

Requirements Management (RM) can be considered from two perspectives:

1) managing requirements and requirement artifacts in the development process
2) managing Requirements Engineering activities (i.e. RM as process management)[Pohl 2010, chapter 30.1]

RM can be used in the context of a development project, in the evolution of an existing system, in software product management or in the continuous, cross-project management of requirements.

The IREB Glossary [IREB Glossary] defines RM as a process to manage existing requirements and related artifacts. This includes storing, modifying and tracking requirements and other artifacts. Among other things, this includes structuring, preparing, consistently changing and implementing requirements [Rupp & Sophist 2009].

Different definitions of Requirements Engineering (RE) and Requirements Management (RM) are often found in the literature. Depending on the definition

- Requirements Engineering is a part of Requirements Management (e.g. in[Schienmann 2001]), or
- Requirements Management is a part of Requirements Engineering (e.g. in [IREB FL 2012]), or
- Requirements Engineering and Requirements Management are defined as co-existing aspects (e.g.[CMMI 2011]).

IREB defines Requirements Management **as a part** of Requirements Engineering.

Within the framework of this syllabus, the terms are therefore defined as follows:

- The term **Requirements Engineering** covers the areas of determination, documentation, checking/reconciliation and the management of requirements.
- The term **Requirements Management**, on the other hand, refers to the task of managing requirements within Requirements Engineering. "Administration" is here equated with "managing".

EU 1.2 Tasks in Requirements Management (K1)

The task of Requirements Management is to provide information about requirements in such a way that other roles in the project can work efficiently with them. In addition, rules and methods must be provided to enable delivery [Rupp & Sophist 2009].

If requirements are collected, three basic assumptions apply on the basis of which all tasks of Requirements Management can be derived and with which the following tasks and methods can be justified (see also[Rupp & Sophist 2009]):

- Requirements must be utilized by many people,
- Requirements change,
- Requirements are supposed to be reused.

The following tasks and concepts are part of Requirements Management [Rupp & Pohl 2011]:

- **Assigning attributes**: attributes make it possible to describe requirements and their meta information in a more structured way, to group them or to make them comparable with other requirements. Attributed requirements are the basis for creating views of requirements.
- **Evaluation and prioritisation**: priorities make it possible to differentiate between important and less important, complex and less complex requirements with the aid of evaluation and prioritisation criteria. This distinction in turn serves as a basis for project management and release planning decisions.
- **Traceability**: traceability makes it possible to trace a requirement throughout the entire life cycle of the system. On the basis of this information, the dependent requirements and other development artifacts can be identified, for example, when a requirement is changed.
- **Versioning**: versioning makes it possible to trace changes to requirements within their life cycle.
- **Reporting**: reporting is the collection, evaluation and presentation of information about requirements or the Requirements Engineering process (briefly: RE process). The information contained in reports serves not only as pure information but also as a basis for project decisions and for controlling the RE process.

- **Process management**: management of Requirements Engineering work processes makes it possible to design the processes of the team efficiently.

Requirements Management tasks are planned and performed by the Requirements Engineer or the Requirements Manager.

EU 1.3 Goals and benefits of Requirements Management (K1)

The goal of Requirements Management is to manage requirements and other artifacts related to requirements in such a way that the requirements can be systematically scanned, grouped, evaluated, changed and tracked with reasonable effort. Among other things, it provides answers to the following questions:

- Which requirements are part of the system? (Assigning attributes)
- Which requirements come from which source? (Assigning attributes and traceabiity)
- Which requirements are urgent and important? (Evaluation and prioritisation)
- Which requirement generates too high costs with too few benefits? (Evaluation and prioritisation)
- Which (sub-)system requirements belongs to which customer requirements? (Traceability)
- Which requirements are part of my systems/products? (Traceability)
- Which version of the requirement was implemented in my system? (Versioning)
- Who has changed the requirement last? (Versioning)
- Which key metrics (key performance indicators) can be used to control my Requirements Engineering? (Reporting)
- How efficiently does my Requirements Engineering work? (Process management)

The importance of Requirements Management within the development process is closely related to the boundary conditions of the project [Rupp & Sophist 2009]. Requirements Management becomes even more important...

- ... the greater the number of requirements,
- ... the longer the estimated lifetime of the product,
- ... the more changes that are expected,
- ... the larger the number of participants in the RE process,
- ... the more difficult it is to reach or involve the stakeholders,
- ... the higher the quality demands on the system,
- ... the more re-usage that is to be carried out,
- ... the more complex the development process,
- ... the more inhomogeneous stakeholders' opinions are,
- ... the more releases that will be developed,
- ... the more important the use of standards is.

Good Requirements Management ...[Rupp & Sophist 2009]

- ... increases the quality of requirements, products and processes,
- ... reduces project costs and project duration,
- ... makes it easier to monitor complex projects during all phases,
- ... improves communication within and among teams,
- ... increases customer satisfaction,
- ... reduces the project risk.

EU 1.4 Requirements Management Plan (K1)

Similar to a project management guide, the Requirements Management Plan (RMP) describes the specifications for the implementation of Requirements Engineering. This includes planning what types of requirements are to be documented, how requirements are managed and who has which responsibilities in the RE process.

The Requirements Management Plan covers in total:

- the Requirements Information Model (RIM), which describes the requirements landscape, i.e. the requirement types to be managed and their levels of detail (see EU 2),
- attributes and views of the requirements (see EU 3),
- prioritization criteria and methods (see EU 4),
- guidelines for version management of requirements and requirement artifacts as well as the change process (see EU 5),
- guidelines for managing traceability of requirements (see EU 6),
- guidelines for describing requirement variants (see EU 7),
- Guidelines for reporting on requirements (see EU 8),
- RE process with activities, roles and responsibilities (see EU 9),
- Guidelines for the tools to be used (see EU 11).

In practice, the Requirements Management Plan is often not an independent document, but rather part of the project management guide, the configuration management plan or other specification documents for the development process.

EU 1.5 Relevant standards (K1)

The topic of Requirements Management is addressed in a variety of standards and guidelines in different ways. Some prominent representatives are listed below.

- The CMMI (Version 1.3) [CMM 2011] considers among others the processes "Requirements Development" and "Requirements Management", whereby some of the assigned objectives differ significantly from the definitions in the IREB.

- ISO 9000 / ISO 9001 [ISO 9000] is a standard for quality management in organizations. ISO 9001 defines minimum requirements for a quality management system and describes, for example, requirements for product realization as well as measurement and improvement and thus addresses topics such as identifiability or traceability (see Clause 7.5.3 Identification and Traceability).
- ISO/IEC 12207:2008 and 15288 ("Systems and software engineering - Software life cycle processes" and "Systems engineering - Systems life cycle processes") define the most important processes for systems and software development and thus also describe tasks in the area of Requirements Engineering and Requirements Management.
- IEC 61508 ("Functional safety of safety-related electrical/electronic/programmable electronic systems") deals with the definition of requirements for the functional safety of systems. Particular attention is given to the topic of traceability.
- ISO/IEC/IEEE 29148:2011 ("Systems and software engineering - Life cycle processes - Requirements engineering") describes processes specifically for Requirements Engineering.
- SOX (Sarbanes-Oxley Act)[US Congress 2002], [Nemnich 2006] is a US federal law in response to accounting scandals intended to improve the reliability of reporting by companies listed on the public capital market in the USA. In essence, the Sarbanes-Oxley Act requires us to know who made what changes when, and thus also relates to the core tasks of requirements management.

Due to external constraints (e.g. explicit customer requirements or regulatory specifications), one or more of these standards or guidelines may apply, and the implementation of Requirements Management is therefore also mandatory.

Some of the standards use different definitions of terms and are therefore only partially compatible with each other.

EU 2 Requirements Information Model (K2)

Duration: 2 hours
Terms: Requirement Landscape, Requirement Type, Forms of Presentation, Requirements Information Model (RIM)

Educational Objectives:
EO 2.1.1 Knowing basic classifications of requirements (K1)
EO 2.1.2 Knowing the Twin Peaks model (K1)
EO 2.2 Knowing typical forms of presentation for requirements (K1)
EO 2.3 Mastering and using the Requirements Information Model (K2)

EU 2.1 Fundamentals (K1)

Requirements can be classified according to various criteria:

- Requirement type
- Independence from solution [Solution independence]
- Level of detail or abstraction level of the requirement

The classification by type of requirement is known from the IREB Foundation Level [IREB FL 2012, EU 1][Pohl 2010]:

- Functional requirements
- Quality requirements
- Constraints

The characteristics of the different types of requirements and their further categorization are discussed in detail in [Pohl 2010, Pohl & Rupp 2011].

In specific projects these requirements types are often refined or different names used for them. Detailed classification schemas for requirements can be found for example in [Young 2004][Rupp & Sophist 2009][Wiegers & Beatty 2013][Robertson & Robertson 2014].

In addition, requirements can be classified orthogonally according to their degree of solution independence [Pohl 2010]:

- **High solution independence, e.g. goals**:
  Goals describe the intention of the system without going into the implementation and thus describe the most solution-independent form of a requirement.
- **Medium solution independence, e.g. scenarios**:
  Scenarios describe in an exemplary way possible sequences of interaction to achieve one or more goals. They are described from the user's point of view.

- **Low solution independence, e.g. solution-oriented requirements**:
  Solution-oriented requirements describe the required data, functions, system behavior, states and quality of the system to achieve the goals and to implement the scenarios.

In the documented form [Pohl 2010] also calls these three classes requirements artifact types.

A third classification used to differentiate requirements is the level of detail, also known as the abstraction level. Requirements are often described at several levels of detail: at a low level of detail, i.e. with few details, to provide an overview, and then at a higher level of detail, i.e. with more details, in order to specify the requirements completely and precisely.

Twin-Peaks Model

Requirements form the basis for the system design. However, practice has shown that a waterfall approach, in which the system is designed sequentially after specifying the requirements, often does not correspond to reality [Pohl 2010], [Cleland-Huang et al. 2013]. Instead, it is rather an iterative process in which requirements shape the architecture of a system and vice versa. Figure 1 illustrates this relationship in the so-called Twin-Peaks model [Nuseibeh 2001]. The vertical axis represents the level of detail of the system description, while the horizontal axis represents the increasing orientation from problem description to implementation. The figure shows that more detailed requirement descriptions are developed iteratively in parallel with more detailed system architectures.

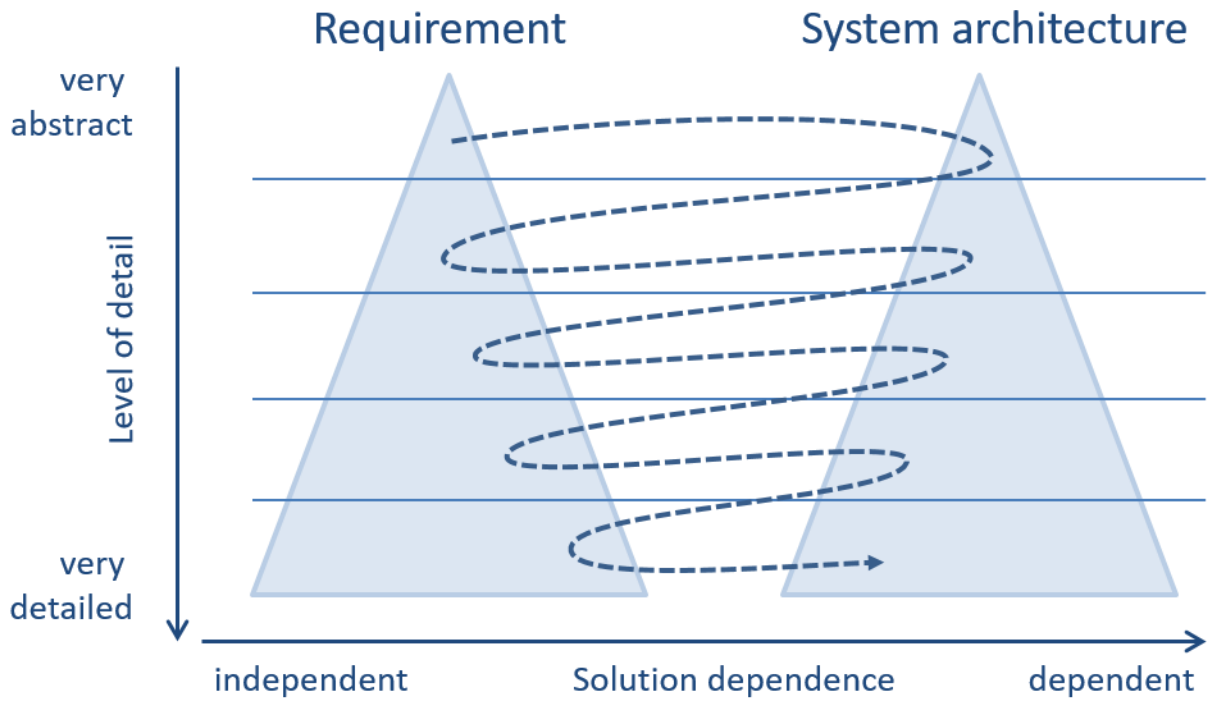


Figure 1: Twin-Peaks model

The definition of the required abstraction layers, i.e. the layers in which the different levels of requirement detail are described, is project-specific and cannot be described universally. However, it is generally recommended to detail requirements until:

- A common understanding of the requirement has been achieved by all stakeholders and it is clear to everyone exactly what is required.
- The remaining degree of latitude for the solution is so small that further clarification would generate more cost than benefit (i.e. the residual risk resulting from remaining degree of latitude is acceptable).
- The requirements are specified to the extent that they are clearly verifiable (testable) against the subsequent solution.

Frequently used abstraction levels are, for example: "overall system", "subsystem", "technical component" or "customer and stakeholder requirements", "system, architecture, component design requirements", "implementation requirements", or also the division into "business specifications" and "technical specifications".

Two aspects of this procedure are crucial for Requirements Management:

- In order to support different perspectives on the requirements, requirements are specified and stored at different abstraction levels at explicit levels of detail.
- In order to later make sense of this incremental and iterative process, and to ensure that completeness and consistency are maintained across different levels of detail, it is essential to link requirements at different levels of detail, but also requirements and their implementing elements in the architecture. This is achieved with traceability relationships (see EU 6).

EU 2.2 Forms of Presentation (K1)

Basically, the three types of requirements, the three types of solution independence and the selected levels of detail result in a number of potential "crossing points" (see Figure 2), hereinafter referred to as classes of requirements.

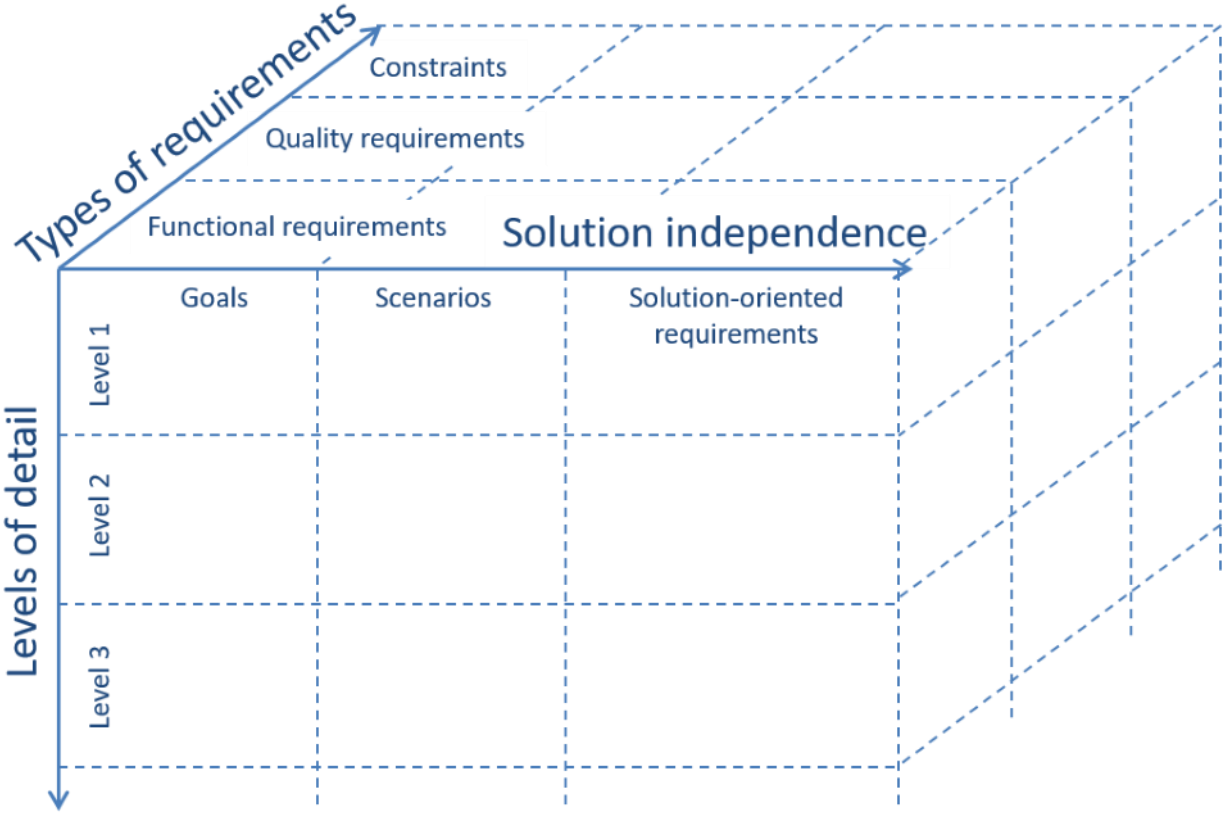


Figure 2: Numerous potential "intersections" (classes of requirements)

It must now be determined (1) which of the "intersections" are to be considered at all and (2) how they are to be documented. A complete examination of all intersections is normally not economically possible and often also not meaningful in terms of content.

For the documentation of requirements there are different forms of presentation, degrees of formality and notations / languages, e.g.:

- Natural language (e.g. description in pure prose)
- Structured text (e.g. templates, patterns)
- Model-based, graphical notation forms (e.g. UML, feature tree)
- Formal descriptions (e.g. mathematical functions)

For each "intersection point" required, it must be specified which form of representation is required. It is also possible that different characteristics are required for a "crossing point", e.g. depending on stakeholder groups or system components. For example, at the "sub-system" level of detail, for type "functional requirement" and with a solution independence of "solution-oriented requirement", it may make sense to use textual requirements (to document general customer requirements) or message sequence diagrams (to document interface requirements).

In addition, the language in which textual and model-based requirements are to be documented must be defined at the very beginning in order to avoid unnecessary additional effort through translation processes later on.

EU 2.3 Creation of a Requirements Information Model (K2)

In real projects, only some of the potential "intersections" resulting from all mathematical combinations of requirement types, solution independence and levels of detail will be considered.

All the "intersections" used in a project are called the requirements landscape. A requirements landscape is a definition of (a) the classification to be used in relation to the requirement types, (b) the classification to be used in relation to the independence of the requirements from solutions, (c) the required levels of detail for each requirement artifact type and (d) the forms of presentation to be used for each requirement artifact type.

In some cases, relevant standards and guidelines already provide a first indication of the "crossing points" to be considered in every case. For example, [ISO 26262] requires that functional safety requirements and the resulting technical safety requirements are derived from safety objectives (i.e. a special form of requirements expressed as objectives).

[Rupp & Sophist 2009] provides an example of five levels of detail, which are referred to as specification levels:

- **Specification level 0**: roughly describes the overall project and its goals.
- **Specification level 1**: describes the use cases and business processes of the business areas that fulfill the goals (functional specification).
- **Specification level 2**: details the business processes and business requirements of the business areas at specification level 1 (functional specification).
- **Specification level 3**: describes detailed user requirements with division into sub-systems and a description of interfaces (functional specification).
- **Specification level 4**: describes the technical specification with a separation into hardware, software and other components (technical specification).

Specification level 0 thus includes all requirement types with solution-independence "goals" and the lowest (i.e. coarsest) level of detail.

This example also shows that not all levels of detail are used for documenting every requirement with regard to requirement type and solution independence.

The definitions of the requirements landscape (i.e. the required "intersections" and the chosen form(s) of representation) should be documented as a Requirements Information Model (RIM). For example, this could be done by creating an entity-relationship diagram or a class diagram. An example can be found on Figure 3. The example is based on the specification levels from [Rupp & Sophist 2009], but does not fully implement them.

In addition to the specifications of requirements type, independence from solutions, abstraction level and form of representation already defined, the requirements information model should be supplemented by further aspects:

- Which attributes will be used where? (see EU 3)
- Which views will be supported? (see EU 3)
- Which evaluation criteria are planned for requirements? (see EU 4)
- Which roles are responsible for maintaining and changing which information? (see EU 5)
- Which traceability relationships amongrequirements and upstream and downstream artifacts will be documented? (see EU 6)
- How will variants of requirements be documented? (see EU 7)

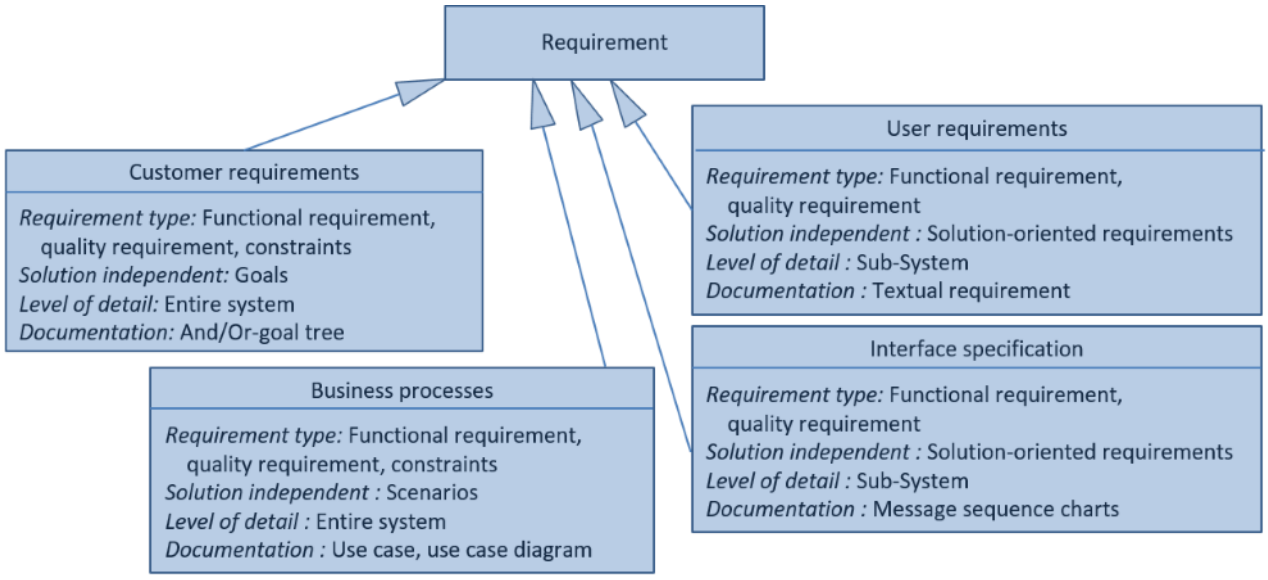


Figure 3: Example of a specific Requirements Information Model (RIM)

The RIM provides all stakeholders of a project with a central view of the requirements artifacts used and their usage in Requirements Management and is therefore an essential part of the Requirements Management Plan (RMP).

When defining the RIM, it is always important to balance the benefits brought by more comprehensive requirements documentation and the costs arising from it [Glinz 2008], [Davis 2005].

When examining a RIM, the following control questions may be used:

- **Formal completeness**: Is it clear for each class of requirements which requirement type and degree of independence from solution it has, to which level of detail it is assigned and with which form(s) of representation it is documented?

- **Content relationships**: Is it clear which levels of detail exist and how they are connected? Is it clear how requirements in the different requirement classes are interconnected?
- **Adequacy**: Are all the requirement classes appropriate to provide sufficiently detailed and complete requirements so that the development activities dependent on can be carried out?

In particular, the question of appropriateness is also strongly dependent on the chosen process model. Agile approaches (cf. EU 10), for example, focus on solution-independent "goals" and "scenarios" and typically omit the (more complex) solution-oriented requirements. Through close contact with the customer, prototyping and short iterations, necessary information can be obtained just in time.

EU 3 Assigning Attributes and Views on Requirements (K2)

Duration: 2 hours
Terms: Attribute, Attribute Schema, View

Educational Objectives:
EO 3.1 Knowing the objective of assigning attributes and the benefits of assigning attributes in management activities (K1)
EO 3.2 Knowing advantages of attribute schemas (K1)
EO 3.3.1 Knowing the steps for designing an attribute schema (K1)
EO 3.3.2 Mastering and using predefined steps to design an attribute schema (K2) …
EO 3.4.1 Knowing types of changes in attribute schemas (K1)
EO 3.4.2 Evaluating modifications to attribute schemas based on the different types of changes (K2)
EO 3.5 Knowing the objectives of creating views on requirements and types of views (K1)
EO 3.6 Knowing the steps of the process for defining views (K1)
EO 3.7 Evaluating the degree of attribute population (K2)

EU 3.1 Objectives of assigning attributes (K1)

Managing the requirements of complex products over their entire lifecycle requires the administration of a large amount of information about those requirements. For example, to determine who submitted a conflicting requirement in the case of a conflict, it is necessary to document the source of this requirement.

The goal of assigning attributes to requirements is to enable stakeholders to document information on requirements in a structured manner as part of the RE process [Pohl 2010, chapter 18.2][IREB FL 2012, EU 8.1]. Therefore different attributes are defined in which this information (e.g. source, creation date, author, etc.) is recorded.

Assigning attributes to requirements provides a basis for a number of tasks in Requirements Management and for other management activities, e.g.:

- **Traceability**: Attributes are used in Requirements Management to realize traceability. A prerequisite, for example, for the traceability of the source of a requirement - in order to be able to request further details, for example - is that the source has been documented in an appropriate attribute. In addition, attributes also facilitate links to other requirements in each case, so that, for example, traceability can be established between goals and solution-oriented requirements.
- **Views**: the conversion of views to requirements is usually based on attributes (see EU EU 3.6).
- **Prioritization**: the respective priority is documented in one or more corresponding attributes. Several attributes can be defined for different prioritization contents.
- **Variant Management**: attributes can be used in variant management (see EU 6) to assign requirements to specific variants.

- **Reporting**: attributes provide a basis for generating reports, such as an evaluation of the number of requirements in a certain status (such as "in progress" or "checked").
- **Project Management**: if the implementation effort per requirement is collected as part of project management, this implementation effort must be recorded in an appropriate attribute. In addition, it is also necessary to document the status of a requirement to support project management.
- **Release Management**: the interface to Release Management is supported by a corresponding "Release" attribute. This allows documenting which requirements are implemented in which release. In many cases, a distinction will be made between the desired and the planned release in order to clarify the differences often occurring between these two releases.

EU 3.2 Use of an Attribute Scheme (K1)

The set of all defined attributes for a requirement type (e.g. functional requirements, quality requirements) is called the attribute schema ([Pohl & Rupp 2011], section 8.1.2). Providing an attribute scheme for requirements leads to the following advantages in Requirements Management [Pohl 2010, chapter 18.2]:

- **Accurate and consistent definition of the required information**: A predefined schema defines which information or attributes for requirements must be entered and which values are allowed for this information.
- **Gap detection**: it is possible to detect gaps in the elaboration of requirements if certain attributes are not populated.
- **Support for employee training**: employees who have already worked with the same, or similar, attribute schemas in a previous project, for example, can quickly find the necessary information and where particular information on the requirement should be documented.
- **Finding the same information in the same place**: as all requirements within a project are documented on the basis of the same attribute schema, it is clearly specified where which information - such as the author - can be found for a requirement.

Typical influencing factors for the definition of an attribute schema are [Pohl & Rupp 2011, chapter 8.1][IREB FL 2012, EU 8.1]:

- Specific characteristics of a project, e.g. project size,
- Guidelines provided by the company, e.g. company standards,
- Properties and regulations in the area of application, e.g. reference models, standards,
- Boundary conditions and restrictions on the development process, e.g. process standards, conventions, etc.

Besides the advantages motivated by the RE process, the goals of other accompanying processes must also be taken into account when defining an attribute schema, for example, for Project Management (see also EU 3.1). The required information must then be mapped using a relevant attribute type for the requirement.

EU 3.3 Designing an Attribute Schema (K2)

The definition of an attribute schema should be made before starting the requirement documentation and should be agreed with all stakeholders of the RE process. Subsequent enhancements and changes are usually only possible with great effort [Rupp & Sophist 2009, chapter 14.1].

The following steps are necessary to design an attribute assignment schema for use in a specific project:

1) **Identification of sources of attributes**

   To select attributes, it is first necessary to identify the relevant sources for attributes. Sources that can be used to select attributes include:

   - An attribute schema from a similar project (for example, similar in scope, number of employees involved etc.),
   - a reference schema of the organization or another standard,
   - organization rules that determine, for example, which attributes must be used in all attribute schemas of all projects,
   - Stakeholders of the RE process.

2) **Selecting Attributes**

   For the successful use of the attribute schema in Requirements Management, it is essential that the attributes are selected purposefully and appropriately for the project. This applies both to the use of reference schemas as well as to the definition of an attribute schema without use of a reference schema.

   Some steps for selecting attributes are listed below:

   - Checking attributes in the reference schema; if necessary, change or supplement the attributes.

- For the selection of attributes for an attribute schema as well as for the evaluation of a reference schema for completeness within the scope of a project, the seven categories presented in [Pohl 2010, chapter 18.3]: "Identifiability", "Context Relations", "Documentation Aspects", "Content Aspects", "Compliance Aspects", "Validation Aspects" and "Management Aspects" can be of assistance.
- Limiting attributes to a pragmatically usable set by focusing on the objectives of Requirements Management in the project [Rupp & Sophist 2009].
- Definition of the context for the uniqueness of the identification number (only unique in the project, unique in the company, etc.)

3) Defining Permitted Attribute Values and Properties of Attributes

After defining which attributes are used in an attribute schema, the properties of the attributes must be defined, such as:

- Which type is the attribute under consideration (text, number, logical value, date enumeration, etc.)?
- Which values for enumeration types should be available for selection in a corresponding attribute?
- Is the attribute a required field?
- Can the attribute contain several values or only one value?
- Are user aids, such as "all values" or "no value", available for selection, and what consequences result from this?

4) Defining Dependencies between Attributes and Their Values

Attributes can be interdependent with regard to their values. For example, it is possible to prevent a requirement with the value "volatile" in the "Stability" attribute simultaneously receiving the value "released" in the "Status" attribute. This ensures that, for example, only requirements that are considered stable are approved for development.

When defining an attribute schema, it should also be determined whether certain combinations are not allowed for two attributes with, for example, predefined attribute values. In this case, it may make sense to combine these two attributes in one attribute and only offer the permitted combinations there. This is particularly useful if the tool used does not support the analysis of dependencies between attribute values.

In Variant Management, assigning requirements to specific variants can be prohibited. It is also possible that certain combinations of variants are not permitted.

Dependencies between attributes and their values can also arise through the hierarchization of requirements. For example, if requirement A is detailed by requirements A.1 and A.2, it must be determined for the attribution schema attributes whether or not the value of A depends on the values of A.1 and A.2.

5) **Providing support for recording data**

The collection of additional information on the actual requirement represents additional effort for the Requirements Engineer, which in some cases benefits other stakeholders (e.g. project managers) rather than himself. This is one of the reasons why it is very helpful for the Requirements Engineer to be supported as far as possible in collecting such information.

This kind of support is provided mainly (or even exclusively) by use of an appropriate tool for recording and managing requirements. A simple example of this is the definition of default values for attributes.

6) **Documenting the Attribute Schema**

Attribute schemas are defined in a tabular form or in an information model, depending on the degree of complexity (for example, with regard to the number of attributes, dependencies between attributes, or their values) (see EU 2). The definition in an information model is used by a tool that maps the corresponding attribute schema and supports the Requirements Engineer in documenting the associated information in a project.

EU 3.4 Change Management of Attribute Schemas (K2)

Subsequent changes to an attribute schema during the course of the project should be avoided if possible [Rupp & Sophist 2009, chapter 14.1].

The consequences of a later change to an attribute schema depend on the type of change:

- **Adding, changing or deleting an attribute:**
  When adding a new attribute, it is necessary to evaluate how time-consuming it is to maintain the previously documented requirements in terms of the new attribute. If an attribute is to be deleted, this can have negative consequences if, for example, modules query the attribute in a tool. Instead of deleting it, one can add "(no longer used)" to its name.
- **Adding, changing or deleting the possible attribute values (value range):**
  Adding attribute values is usually no problem for the underlying tool. From a technical point of view, it must be examined whether the addition results in a new meaning for the existing values of an attribute, so that requirements previously evaluated against this attribute have to be analyzed again and, if necessary, new values set. When deleting attribute values from a value range, it is important to ensure that requirements do not become inconsistent due to empty entries. Problems are mainly caused by mandatory fields, since the requirement must have a value in the attribute under consideration. In this case, a solution may be to enter a default value in the field.

  When attribute values are changed, it is important to ensure that the changes are made in all requirements that contain the original value. In general, when changing or deleting requirement values, it is important to decide whether this change will affect requirements that have already been entered or only applies to future requirements.

  For dependent attributes with dependent attribute values, it has to be ensured that the removal of an attribute value does not result in inconsistent data.

- **Adding or deleting relationships between attributes:**
  Adding a relationship can lead to inconsistencies within the requirement set already captured. For example, if a definition is added such that selecting a value for the attribute "Stability" should now always lead to the attribute "Risk" being populated (and is therefore always populated when "Stability" is populated), then requirements with a value for "Stability" but none for "Risk" must be checked again.

  If the dependency between attributes is removed, you must ensure that default values (in the sense of: If attribute value A is selected, then, for example, attribute value B is automatically predefined by a tool) will be removed as well.
- **Changing default values for attribute type:**
  Changing default values should initially only affect the entry of new requirements. In this context, however, requirements that have been assigned the previous default value should be analysed to check if they still have the correct value, or if they must be adjusted.
- **Changing the property of "Mandatory fields" and "Optional fields":**
  Changing a mandatory field to an optional field usually does not result in any subsequent effort. If a change from an optional attribute to a mandatory attribute is planned, then it must be ensured that the attribute is populated with an appropriate value for all already documented requirements. It may be necessary to assign custom values rather than using a default value.

Generally, when making changes to attribute schemas, the extent to which the underlying tools are affected must be analyzed. For example, if scripts have been created in the DOORS tool that check or process a particular attribute, a corresponding change to the attribute can result in the scripts no longer being executable.

EU 3.5 Goals and Types of Views (K1)

An advantage of defining attribute schemas is the possibility of defining views for the information model and implementing them as part of tool support (see also [IREB FL 2012, EU 8.2]).

In a project, the complexity of the information on the requirements can be very high, especially where there are a large number of requirements to be documented. In addition, many stakeholders are involved in the project in regard to requirements. In addition to the Requirements Engineer, these are, for example, developers, testers or project managers.

The aim of providing specific views is to provide the respective stakeholders with targeted and role-specific information about the requirements and thus reduce complexity from the perspective of the specific stakeholder. In addition, access to requirements can be controlled on a role-specific basis by creating views (see also [Pohl 2010, Chap. 18.5]).

Views can be classified as follows:

- **Selective Views**: Selective views limit the number of requirements to be considered using a predefined filter.

- **Projective Views**: Projective views hide attributes that are not relevant for the considered view.
- **Condensed Views**: Summarizing views enable evaluations (for example, creation of totals) of a requirements set with reference to certain attributes.
- **Combination of selective, projective and condensed views**: In combined views, certain attributes are hidden, the requirements set is restricted with regard to one or more criteria and evaluated separately using an attribute, for example.

Tools that provide views of requirements usually also have the ability to sort by different attributes [Rupp & Sophist 2009, chapter 14.1]. This allows stakeholders in their respective roles to change the focus on requirements (for example, all requirements with a high implementation effort at the beginning of a list) without hiding requirements. In addition, tools often allow a further structuring of views by grouping requirements according to an attribute.

EU 3.6 Definition of Views and Risks of Views (C1)

The process for defining views includes the following steps:

- **Determining the stakeholders who need one or more views:**
  Views can provide different perspectives on requirements. To define the necessary views, all stakeholders who will use views on the requirements have to be defined first. The stakeholders then serve as a source for defining a view.
- **Reusing**: Views from other projects or from a reference project can also be used as a source of views to be defined.
- **Determination of the goals of a view in relation to the different stakeholders:**
  Depending on the respective stakeholders, it must be determined which goal will be pursued with the specific view. This can be used, for example, to determine whether consolidation is necessary or which sorting should be set initially. In this context, stakeholder rights and views must also be determined, i.e. which stakeholder should be able to activate which view. In doing so it makes sense that one view can be used by many stakeholders.
- **Determination of necessary attributes and comparison with the attribute schema:**
  To be able to fulfill the goals of a view, it must be ensured that it is possible to collect the necessary information and that the corresponding attributes are also available. However, the comparison with the attribute schema often leads to new views being generated, because only when the attribute schema is closely examined do the stakeholders realize, for example, which evaluations would still be possible. The definition of views and attribute schemas is therefore a highly integrative process; both processes influence each other.
- **Implementing the view:**
  Finally, the predefined views must be implemented and tested in the underlying tool.

The creation of views brings some risks. In many cases, users of a Requirements Management system are not aware that the large amount of information on a requirement can be restricted by views as required.

They then often work with a global and all-encompassing view and perceive the tool as too extensive and possibly obstructive in their work. There is also the danger of defining views in which too much context is lost. For example, if one creates a view in which atomic requirements are given in a list without any context (e.g. use cases), this overview will only be meaningful to a limited extent. In order to avoid such ineffective views as far as possible, it is very important to consider the underlying goal of a view.

EU 3.7 Optimizing Attribute Population and the Creation of Views (K2)

Practice shows that in some projects the attributes are not populated to the expected extent. Therefore it is recommended to check regularly and at specific points in a project whether, and how, an attribute is to be used [Rupp & Sophist 2009, chapter 14.1].

One way of ensuring the degree attribute population in the context of tool usage is to define the attribute as a "mandatory field". This forces input when a requirement is created. In this context, it should be noted that the definition of too many mandatory fields can greatly impede processing, as, for example, information may not yet be available at the time a requirement is first entered. For this reason, mandatory fields should only be declared as such sparingly and with a sense of proportion.

A prerequisite for checking the capture of attributes is a corresponding evaluation, which can usually be carried out within the scope of a tool used.

For optional attributes that are not necessarily required to be entered, the following consequences can result from their evaluation:

- **The attribute was not used in either a view or a report**: This case indicates that the attribute in question does not have a specific target, since it is not visible in any view or analysis.
- **The attribute is always populated with the same value, for example the default value**: In this case, there does not seem to be any real distinction in the context of the attribute for the different requirements, which suggests that the proposed list of values is not suitable. The Requirements Engineer should check whether either the attribute can be omitted (because there is no real target for it) or the selection list must be adjusted. In the latter case, the notes from EU 3.4 should be taken into account.
- **The attribute is never filled**: If the attribute has deliberately not been filled, the information may not be important. If this assumption is confirmed, the attribute should be removed. However, the reason often lies in the fact that users are not aware of the intention of the attribute or do not directly see any benefit since the information is used by another stakeholder. Then the users should be (re)trained to explain to them the added value of the particular attribute. In this case, the attribute can be used again.

- **The attribute is only filled for a few requirements**: The question here is whether the goal associated with the attribute can be achieved, or whether it is still relevant at all. If this is not the case, the attribute can be removed. However, if it turns out that the attribute is important, it can be declared as a mandatory field, which forces entry in the future. In this case, requirements which have no value in the attribute under consideration must be updated retrospectively (for example, automatic population with a standard value).
- **The attribute is not populated in individual cases**: It must first be determined whether this attribute is still relevant for the project. If yes, the Requirements Engineer should complete the relevant requirements. If the attribute is no longer considered essential, it can either be removed or the gaps can be tolerated.
- **The attribute is always populated**: In this case, no further activity is necessary.

If the feedback from the stakeholders of the RE process results in new requirements with regard to the defined and implemented views, these should be examined, prioritized and, where necessary, implemented. Such requirements for views may lead to Change Management of the attribute schema, if, for example, an additional attribute or an additional attribute value is required (see EU 3.4).

EU 4 Evaluation and Prioritization of Requirements (K2)

Duration: 1,75 hours
Terms: Prioritization, Evaluation, Prioritization Techniques, Ad-Hoc Prioritization Techniques, Analytical Prioritization Techniques

Educational Objectives:
EO 4.1 Knowing the basic characteristics of an evaluation (K1)
EO 4.2.1 Knowing the relationship between evaluation and prioritisation (K1)
EO 4.2.2 Knowing the basic prioritization procedure (K1)
EO 4.3 Mastering and using ad hoc prioritization techniques (K2)
EO 4.4 Mastering and using analytical prioritization techniques (K2)
EO 4.5 Knowing the procedure for combining prioritization techniques (K1)

EU 4.1 Principles of Evaluation (K1)

In all activities of Requirements Engineering, requirements are evaluateed on the basis of various criteria. Thus, during elicitation for example, requirements are categorized according to the Kano criteria (see [IREB FL 2012, EU 8.3]), while during documentation they are differentiated according to their legally-binding nature, or requirements are evaluated according to their criticality because of guidelines and standards to be followed (such as [ISO 26262]).

The task of Requirements Management is to document these evaluations in an appropriate form (see EU 3) and to foresee the consequences of the evaluation for the RE process (see EU 9).

For example, the evaluation of a requirement can have an effect on the further handling of the evaluateed requirement. A requirement that has been evaluateed as particularly safety-critical could, for example, be subjected to more detailed Quality Assurance during testing than a requirement that is considered non-critical.

Possible evaluation criteria are for example:

- The legally-binding nature of a requirement,
- Implementation effort/costs,
- Criticality,
- Stability
- Degree of innovation.

Sources for evaluation criteria are e.g.:

- Project Management,
- Guidelines and standards,
- The requirements attribute schema,
- The subsequent development disciplines.

It is important to define the following basic characteristics of each evaluation when defining the required evaluation criteria:

- Value ranges of the evaluation (e.g. "low", "medium" and "high" for the evaluation criterion "Criticality "),
- The group of people who may carry out the evaluation (for example "architectural relevance" should only be evaluated by architects),
- The earliest time for the evaluation (e.g. the implementation effort may only be estimated when a detailed design is available),
- The latest time for the evaluation (for example, the implementation effort must be estimated before a requirement can be assigned to a release or a development iteration).

Evaluations that have already been performed can then be used in the course of the project to make decisions related to all Requirements Engineering activities [Pohl 2010]. Thus, evaluations can serve as a basis for consolidation using analytical consolidation techniques (see [IREB FL 2012]) or as a basis for prioritizing requirements.

EU 4.2 Prioritizing Requirements (K1)

There is no project where the available resources are unlimited. To ensure that the available resources are used in a goal-oriented manner, it is therefore essential to prioritize the existing requirements. In principle, prioritization should always follow the following process:

- Determining the goals of prioritization
- Determining the prioritization criteria
- Determining the prioritizing stakeholders
- Determining the requirements to be prioritized
- Selecting the prioritization technique
- Adjusting the attribute schema, if necessary
- Performing the prioritization.
- Regular checking and, if necessary, reprioritising of requirements

The prioritization criteria required depend on the goal of the prioritization. Therefore, the first step is to determine which decisions are to be made on the basis of prioritization. Thus, when deciding which functions of the system should be specified in detail first, other prioritization criteria must be considered as compared to when determining the order of implementation. For this reason, it makes sense to determine at the beginning of the project which prioritizations will be required during the course of the project and which evaluations must be carried out before the prioritization.

Depending on the pursued goal, it is necessary to determine which evaluation or prioritization criterion, or which combination of criteria, is used to determine the requirement's priority.

When selecting requirements to be prioritized, it should be noted that the requirements should be at the same level of detail to avoid distorting the result of the prioritization [Wiegers & Beatty 2013].

EU 4.3 Ad-Hoc Prioritization Techniques (K2)

There are a number of techniques for prioritizing requirements, which sometimes differ greatly in terms of the effort required for prioritizing and the suitability for certain prioritization situations. "In projects, simple ad-hoc prioritization techniques [...] are a pragmatic [...] approach for effective prioritization of requirements" [Pohl 2010]. The following ad-hoc prioritization techniques work well in practice:

- Requirements Triage [Davis 2003]
- Ranking [IREB FL 2012, LE8.3] [Pohl & Rupp 2011]
- Top-Ten Technique [IREB FL 2012, EU8.3][Pohl & Rupp 2011]
- Single Criterion Classification [IREB FL 2012, EU 8.3][Pohl & Rupp 2011]
- Planning Poker
- Two-Criteria Classification
- 100-Dollar Technique
- Kano classification [IREB FL 2012, EU 8.3] [Pohl & Rupp 2011]

EU 4.3.1Two-Criteria Classification (K2)

If the result of a single criterion classification is not differentiating enough, for example, because several prioritization criteria have to be considered or because too many requirements lie within the same specification of the selected criterion, the values of several criteria can be multiplied and the results prioritized via a ranking. Each requirement is now assigned to a possible combination of values and thus receives the prioritization assigned to this combination. This procedure is usually presented in the form of matrices. A sample matrix with the prioritization criteria *Customer Benefit* and *Costs* is shown in Figure 4.

The Eisenhower principle, named after the former US president, is also very popular and proposes a classification based on criteria of *Importance* and *Urgency* (see Figure 5).

		Cost		
		High	**Medium**	**Low**
Customer Benefit	**High**	Priority 5	Priority 2	Priority 1
	Medium	Priority 6	Priority 4	Priority 3
	Low	Priority 9	Priority 8	Priority 7

Figure 4: Sample Two-Criteria Classification

		Urgency	
		Urgent	**Not urgent**
Importance	**Important**	Priority 1	Priority 2
	Not important	Priority 3	Priority 4

Figure 5: Two-Criteria Classification by Importance and Urgency

EU 4.3.2100-Dollar Technique (K2)

The 100-Dollar technique is well suited for prioritizing a few requirements. The use of this technique is recommended for coarser requirements at a higher level of detail or for requirement clusters.

Stakeholders are granted 100 imaginary units (money, time, etc.), which they can distribute among the requirements. The stakeholders involved have a defined period to reflect on the allocation of resources available to them before allocating these to the requirements. The more units a requirement has received at the end, the higher the priority of this requirement.

Theoretically, the technique also works with larger quantities of units (1000, 10000,...). However, this costs much more time and effort. It is recommended to use a tool (e.g. software) to check the sum of the units allocated per stakeholder.

This technique should only be applied once to a specific set of requirements, as stakeholders could be influenced by the distribution of other stakeholders and thus potentially distribute their units differently the next time.

EU 4.4 Analytical Prioritization Techniques (K2)

In some cases, ad-hoc prioritisation techniques are strongly influenced by the stakeholders involved in the prioritisation process and are therefore only recommended to a limited extent for very critical decisions. More neutral prioritization can be achieved using the following analytical prioritization techniques.

- **Wieger's Prioritization Matrix:** The matrix proposed by Karl Wiegers for prioritizing requirements compares the relative advantage and relative disadvantage of each requirement with the relative costs and relative risk of each requirement. Accordingly, requirements with high customer benefit and at the same time low costs and risks are given higher priority than requirements with comparatively low customer benefit and the same costs and risks [Wiegers & Beatty 2013].
- **Analytical Hierarchy Process (AHP):** The basic idea behind the AHP is the pairwise comparison of all requirements to be prioritized. A scale from 1 to 9 determines how much more important requirement A is compared to requirement B [Karlsson & Ryan 1997].

EU 4.5 Combination of Prioritization Techniques (K1)

The analytical prioritization techniques scale very poorly compared to the ad-hoc prioritization techniques. Thus, for both the Analytical Hierarchy Process and Wiegers' prioritization matrix, a maximum number of 25-30 requirements is recommended for prioritization in order to keep the complexity and the time required for the methods within manageable limits [Wiegers & Beatty 2013][Moisiadis 2002].

Since many projects work with a much higher number of requirements, and detailed prioritization in the lower range of the prioritization scale offers little added value, a combination of ad-hoc and analytical prioritization techniques has proven successful.

For example, when planning a release, if it is necessary to decide which requirements are to be implemented in this release; all requirements will be prioritized ad hoc in a first step. The Requirements Triage [Davis 2003], a Single Criteria Classification inspired by the medical domain, or the Eisenhower Principle, are widely used here.

If the ad-hoc prioritization technique has identified more or less requirements to be implemented than can be implemented in the planned release, requirements close to the acceptance limit are prioritized using an analytical prioritization technique and the requirements with the highest priority are transferred to the release or, conversely, those with the lowest priority are removed.

EU 5 Version and Change Management (K2)

Duration: 2 hours
Terms: Version Control, Requirements Configuration, Requirements Baseline, Release, Requirements Change, Change Management Process

Educational Objectives:
EO 5.1.1 Knowing Version Control activities (K1)
EO 5.1.2 Knowing the characteristics of Requirements Configuration (K1)
EO 5.1.3 Knowing development tasks supported by Requirements Baselines (K1)
EO 5.1.4 Knowing the necessity and disadvantages of requirements branching (K1)
EO 5.2.1 Knowing the main reasons for requirements changes (K1)
EO 5.2.2 Knowing types of requirement changes (K1)
EO 5.2.3 Knowing heuristics for evaluating the stability/volatility of requirements (K1)
EO 5.3.1 Knowing the goal and tasks of a Change Control Board (CCB) (K1)
EO 5.3.2 Mastering and using the Change Management process (K2)

EU 5.1 Versioning of Requirements (K1)

EU 5.1.1Version Control for Requirements and Requirement Documents (K1)

Requirements versioning is an essential part of Requirements Management [Wiegers & Beatty 2013]. Version Control of requirements refers to the process that enables specific development stages of requirements and requirements documents to be kept available throughout the life cycle of a system or product [IREB FL 2012]. Stakeholders can thus trace the history of requirements and requirements documents and can always refer clearly to a specific status of a requirement or a requirements document. These possibilities are helpful in all projects, but they especially support collaborative projects.

According to [Wiegers & Beatty 2013], Version Control consists of three main activities:

1) Defining a schema for identifying versions
2) Identifying versions of individual requirements
3) Identifying requirement artifacts

In the same project, requirements can be versioned both at the level of individual requirements (see activity 2) and at the level of requirement artifacts (see activity 3). Version Control at the document level only allows a rough proof of changes, while at the requirements level every change to a requirement is precisely traceable. Therefore, Version Control at requirement level is much more complex than at document level.

Versions of a requirement or requirements document must be uniquely identified. For this purpose, a corresponding labelling scheme must be defined and used throughout each project.

Every participant of the project must have access to the current version of the requirements. Changes must be clearly documented and communicated to the people affected [Wiegers & Beatty 2013].

When creating a new version of a requirement or requirements document, it is important to record the responsible project participant, the time of the change and the reason for the change [Rupp & Sophist 2009].

EU 5.1.2 Requirements Configuration (K1)

A requirement configuration summarizes a consistent set of logically related requirements or requirement artifacts [IREB Glossary], where each requirement and each artifact is available in at most one version in the configuration. According to [Pohl 2010], a configuration has the following properties:

- **Consistency** - combined requirements and requirements documents are consistent and logically belong together.[1]
- **Uniqueness** - a configuration has an identifier that uniquely identifies it.
- **Unchangeability** - changes to individual requirements or requirements documents of a configuration lead to a new version of a configuration.

EU 5.1.3 Requirements Baseline (K1)

A Requirements Baseline is a consolidated requirements configuration that is stable in terms of content [IREB Glossary], [Pohl 2010].

The baseline should therefore only contain requirements planned for a particular version of the product (e.g. release) and not those that are proposed or still in progress [Wiegers & Beatty 2013].

This creates an unambiguous basis for communication for all project participants. Version Control makes it possible to link the current status of selected requirements with a baseline and later reconstruct it unambiguously.

According to [Rupp & Sophist 2009] and [Pohl 2010], requirements baselines support three essential activities in the development process:

- They provide the basis **for planning releases**. As configurations of stable product requirements that are visible for the customer, they serve as a basis for discussion when defining a release.
- They are used to **estimate the implementation costs** of a particular release.
- They enable a **comparison with competing products** on the market.

A suitable time to create a requirement baseline could be the achievement of a milestone, the completion of a subsystem specification, or a system release [Rupp & Sophist 2009].

[1] In practice, configurations that are not consistent in content are often created. Such configurations are built out of the need to freeze the current work status in order to be able to access it later if necessary. For example, a configuration can be created that documents the starting point of review activities.

EU 5.1.4Branching of Requirements (K1)

A requirements branch describes a set of requirements that have been copied from the current requirements configuration at a certain point in time and changed independently of the original since that point in time. Unlike versions, requirements are valid in different branches at the same time [Rupp & Sophist 2009]. Requirement branches are created, for example, if customer-specific variants of a requirement specification must be created for a product.

Managing requirements is complicated and causes more harm than good when done in an undisciplined manner [Rupp & Sophist 2009]. In particular, the following problems occur:

- The clear identification of requirements will become more difficult.
- Besides versions and improvements, branching forms a third dimension of requirements development and thus increases the complexity of Requirements Management.
- Branches generate redundant requirements information, which must be maintained in parallel and then merged again in the long term.

EU 5.2 Change Management for Requirements (K1)

EU 5.2.1Causes, Sources and Timing of Requirement Changes (K1)

Requirements for a (software) system are subject to changes during the life cycle of this (software) system. These changes may be necessary for several reasons:

- New or changed stakeholder needs [Pohl 2010, chapter 37.1], [Van Lamsweerde 2009, p. 222]
- Changes in the system context (e.g. legislative changes)[Pohl 2010, chapter 37.1], [Van Lamsweerde 2009, p. 222]
- Errors in existing requirements
- Impact of changing a requirement on other dependent requirements
- Architecture and implementation decisions that have repercussions on requirements

Changes to requirements can occur during the entire development and service life of a system.

The complexity resulting from the different points in time at which changes may occur requires both a well-defined process and suitable methods and tools.

EU 5.2.2Types of Changes to Requirements (K1)

Changes to requirements are classified as follows:

- Integrating a new requirement
- Deleting an existing requirement
- Changing a requirement

Changes comprise not only the direct description of the requirement, but also other attributes (for example, evaluation of stability) or relationships to other artifacts (for example, to use cases or other requirements).

EU 5.2.3 Analysis and Documentation of the Stability of Requirements (K1)

Requirements can be classified with regard to their stability, and thus with regard to the probability of the current version of the requirement being changed. Unstable requirements will be affected more by changes than stable requirements. Such a classification can be performed by the evaluation of an attribute provided for this purpose (see EU 3). This evaluation is used, for example, to evaluate the risk of implementing a specified requirement. A specific evaluation can also express the estimation of the effort still required for creating a stable documentation of the requirement. In addition, in the case of major changes to a system, the evaluation can also be used to separately consider requirements that are fundamentally volatile, as they are based on decisions among various alternatives.

The following heuristic rules serve to detect probable changes or unstable requirements and to keep a focus on them (cf. also [Van Lamsweerde 2009, p. 224]).

- Requirement groups that serve the same goal and are generally highly stable (measured by the frequency of changes) have a lower likelihood of change than individual requirements.
- Goals and conceptual aspects are more stable than solution-oriented requirements.
- Functional requirements that meet the core goals are more stable than quality requirements.
- Functional requirements that repeatedly appear in the set of requirements (as amalgamations, extensions or variants) are usually considered as stable requirements.
- Requirements describing alternative choices should be handled with particular caution and are generally less stable than the above, as decisions are often based on incomplete knowledge and assumptions.
- Requirements that are assigned to a variant or enhancement of the system are more stable than requirements that have not yet been assigned.
- Requirements that were frequently changed before are unlikely to be stable.
- Requirements that belong to a group of mostly stable requirements are more stable than those that belong to a group of mostly volatile requirements.

EU 5.3 Change Management Process (K2)

The goal of Change Management is to pre-check each change to one or more requirements in order to control the associated risk and to be able to trace each change later on. The Change Management process achieves this by defining activities, responsibilities and necessary artifacts that describe a clear procedure for handling change requests for requirements.

The Change Control Board (CCB) plays a central role in the Change Management process. To be able to carry out its task, namely the evaluation of changes, in the best possible way, it makes sense to fill the board with roles from various areas, e.g. [Pohl & Rupp 2011, chapter 8.5].

- Customer
- Architect
- Requirements Engineer
- Developer
- Tester

It is essential that at least one representative from each side - the customer and the supplier - is involved.

The main activities of the Change Management process can be described as follows. The input for a Change Management process is a change request, which is described in a predefined form (template) for the process.

- **Step 1**: Preparing the change request
- **Step 2**: Formal evaluation of the change request
- **Step 3**: While classifying a change request it is determined whether it is a corrective, adaptive or exceptional change [IREB FL 2012, EU 8.6]. The Requirements Engineer is involved in the evaluation, to determine, for example, the cause of a change. This evaluation is for example important to determine whether a change has been triggered by the vendor of the (software) system or by the customer.
- **Step 4**: The goal of impact analysis is to estimate and document the consequences of changes. These consequences must be evaluateed not only for other requirements, but also for other artifacts (architecture, source code, test cases, training materials).
- **Step 5**: The results of the impact analysis are used by the Change Control Board to determine whether to approve or reject the change request. It is not always reasonable to accept and implement a change request. Reasons for a possible rejection of a change request are, for example:
  - The change is too costly and is not justified in relation to the effort required for its implementation or its expected benefit.
  - The desired change contradicts other requirements.
  - Implementation of the change would lead to too high a risk with regard to the stability of the (software) system under consideration.

- The change is not covered by a contract.

For reasons of traceability and of achieving agreement among the stakeholders involved, it is essential to document the decisions of a Change Control Board.

- **Step 6:**Accepted change requests are prioritized by the Change Control Board.
- **Step 7**: Accepted change requests are planned for implementation and implemented.

EU 6 Requirements Traceability (K2)

Duration: 2 ½ hours
Terms: Artifact, Requirements Artifact, Traceability, Traceability Model

Educational Objectives:
EO 6.1 Knowing the reasons for requirements traceability (K1)
EO 6.2 Knowing various traceability views (K1)
EO 6.3 Knowing relationship types for traceability relationships (K1)
EO 6.4 Knowing forms of presentation for traceability relationships (K1)
EO 6.5 Mastering and using a specific traceability strategy (K2)
EO 6.6 Mastering and using a specific traceability model (K2)
EO 6.7 Knowing measures for evaluating implemented traceability (K1)
EO 6.8 Knowing the challenges in tracking non-textual artifacts (K1)

EU 6.1 Reasons for Requirements Traceability (K1)

Traceability of requirements is essential for Requirements Management. In the context of Requirements Management, the implementation of traceability basically refers to the maintenance of relationships between different requirements and other development or quality assurance artifacts.

The goal-oriented maintenance of traceability relationships allows existing dependencies between artifacts to be known, for example to prove the implementation of requirements or to identify which changes result from customization of a certain requirement.

In the subsequent sections of this learning unit, we recap the concept of traceability before then explaining its benefits.

In the first step the different terms for requirements traceability are explained. In the literature there are different terms which essentially mean the same thing: tracing, verifiability, traceability, requirements traceability, etc. In this learning unit we will use the term **traceability** unless we refer to a specific reference in the literature.

EU 6.1.1 What does requirements traceability mean? (K1)

By requirements traceability we understand the ability to trace dependencies from requirements to other artifacts throughout their development- or life-cycle. The information to be documented for traceability is defined by the objective to be achieved through traceability (see 0).

For example, if traceability is used to ensure that all business requirements in a project are covered by system requirements, or conversely that a system requirement serves at least one business requirement, then a simple bidirectional reference between these artifacts may be sufficient.

EU 6.1.2 Why Traceability of Requirements and Artifacts is Important (K1)

Traceability of requirements and other artifacts is usually not a project goal, but rather a means to an objective, for example to prove whether and how a requirement has been implemented and tested. A number of reasons motivating traceability between artifacts can be found in the literature, see [Hull et al. 2011][Pohl & Rupp 2011][Wiegers & Beatty 2013]:

- Demonstrability of how goals and requirements are to be achieved
- Verifiability as to why, if and how a requirement was implemented
- Identification of unnecessary requirements and properties of the system (gold plated solutions)
- Identification of missing artifacts (e.g. missing test cases)
- Simplification of assignment of development efforts to requirements
- Support for reusability of artifacts
- Support for maintenance, admnistration and further development of systems

Requirements traceability also helps to answer important questions, such as the impact of a change or why a requirement even exists. In particular, the following analyses are significantly simplified by the presence of traceability relationships cf: [Hull et al. 2011, p. 11ff.][Ebert 2012, p. 305 ff.][PMI 2013]:

- **Impact analysis**: Analysis of which artifacts are affected by a change (reduction or extension of scope) (see EU 5.3 Change Management Process).
- **Source analysis**: Analysis of why a certain artifact (e.g. requirement) exists in order to identify and avoid unnecessary requirements, for example.
- **Coverage analysis**: Analysis of whether all requirements and subsequent development artifacts were considered so that the desired product can be completely scoped, developed and tested.
- **Earned Value Analysis**: Analysis to determine work progress (performance value), in order to compare it against the original project plan and, if necessary, take appropriate action.

Furthermore, traceability between requirements and other artifacts is necessary to meet certain maturity levels for reference models (e.g. CMMI) or legal constraints (e.g. ISO 12207).

EU 6.2 Different Traceability Views (K1)

Requirements traceability can essentially be distinguished by the following dimensions:

- **Traceability among requirements at the same level of detail**. This type of traceability describes, for example, content-related dependencies between functional requirements.
- **Traceability among requirements at different levels of detail**. This type of traceability describes, for example, the detailling of legal requirements into system requirements.
- **Traceability between versions of requirements**: This type describes the traceability of the evolution of a requirement over time. A particularity of this view is that there is only one valid version at a given time.
- **Traceability among requirements and downstream development artifacts**. This type of traceability describes, for example, dependencies that document the implementation / realization of a requirement as a system component or test case.
- **Traceability among requirements and upstream artifacts**. This type of traceability describes the justification or source of a requirement.

The last two dimensions can often be found in specialist literature under the term **Pre- and Post-Requirements Specification Traceabi**lity [Gotel & Finkelstein 1994] or the **extended Pre- and Post-RS Traceability** [Pohl 2010][Pohl & Rupp 2011].

Pre- and Post-Requirements Specification Traceability

- **Pre-Requirements Specification Traceability** is the traceability of requirements to their origin, for example to the upstream goals and visions or other sources of requirements from the system context, such as existing documents and stakeholders.
- **Post-Requirements Specification Traceability** is the traceability of requirements to subsequent development artifacts, such as the architectural design, implementation, test cases.

Advanced Pre- and Post-Requirements Specification Traceability

- In addition to differentiation into pre- and post-requirements specification traceability, traceability among requirements artifacts is also considered. This includes refinements and traceability to dependent functionalities, quality requirements, etc.

Figure 6 graphically illustrates the concept of extended pre- and post-requirements specification traceability described above from the perspective of requirements to upstream and downstream artifacts, as well as traceability between requirements, both on the same and between different abstraction levels.

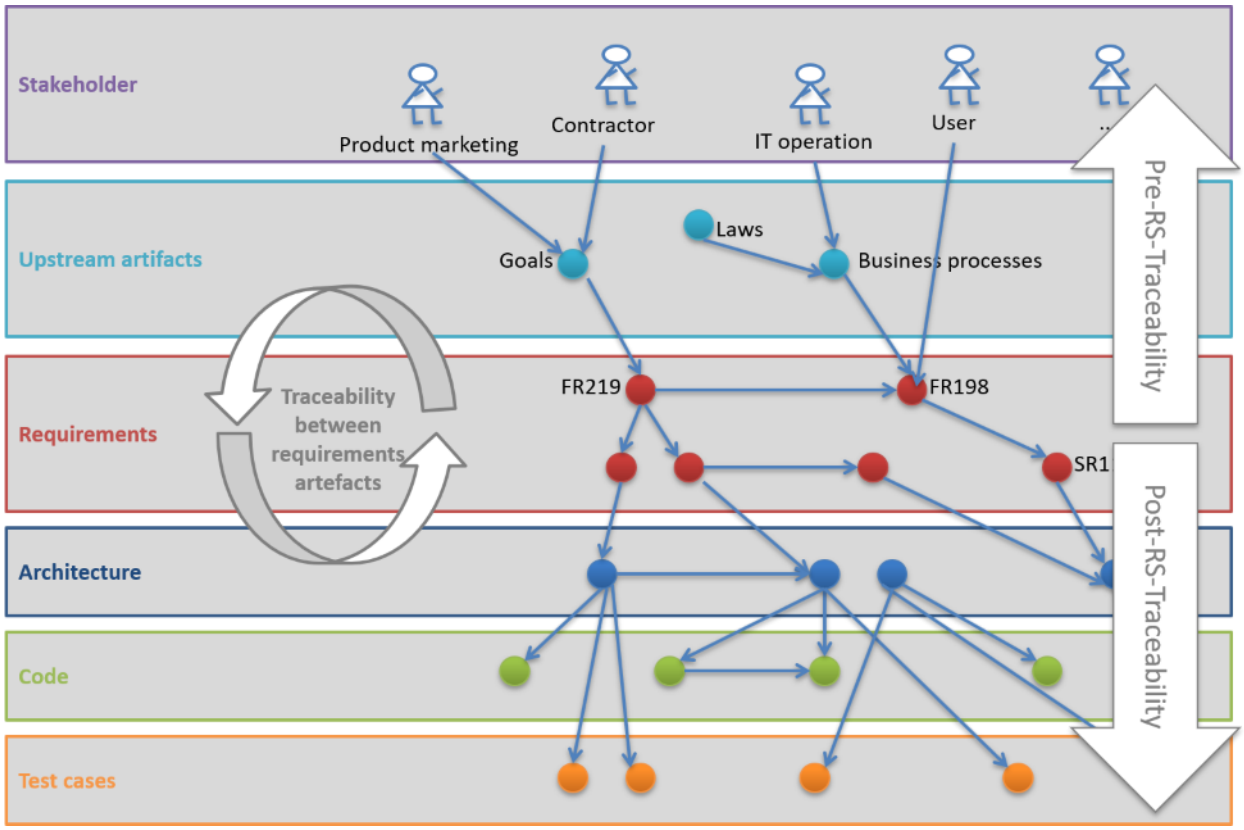


Figure 6: Differentiation of traceability views

It is important to realize that traceability does not begin or end with requirements, but that traceability of requirements should be "holistic", i.e. across all abstraction levels and phases - so from the source (pre-RS traceability) to implementation or acceptance (post-RS traceability).

The granularity of traceability can be at the level of individual requirements, but also at the level of requirement groups or requirements artifacts.

EU 6.3 Relationship Types for Traceability Relationships (K1)

There is no generally accepted definition of types of traceability relationships. It is essential that traceability relationships are used according to the goal of documentation and that there is agreement in the team as to which types of relationships are used, what they mean and among which artifacts they are used.

Theoretically, traceability relationships could be represented by a single type of relationship (e.g. related to). However, this relationship does not say anything about how the artifacts relate to each other. An artifact can be a detailing, a realization, a test case, a variant or even a contradiction of another artifact.

Therefore, different types of traceability relationships can be found in the literature that can be used for specific traceability documentation.

Pohl divides traceability relationships into different classes [Pohl 2010], to which different relationship types can be assigned:

- **Condition**: The class "**condition**" contains traceability relationships that describe content-related dependencies between two artifacts (constraint, precondition, etc.}.

  Example: **Requirement 1** (is_condition_for) **Requirement 12**
- **Content**: The class "**Content**" contains traceability relationships that compare the content of two artifacts (equality, contradiction, conflict, etc.).

  Example: **Requirement 6** (stands_in_contradiction_to) **Requirement 10**
- **Documentation**: The class "**Documentation**" contains traceability relationships that provide further information about an artifact (reason, example, comment, test case, etc.).

  Example: **Test Artifact 99** (is_test_case_for) **Requirement 3**
- **Abstraction**: The class "**abstraction**" includes traceability relationships that describe abstraction relationships between two artifacts (classification, aggregation, generalization, etc.).

  Example: **Requirement 43** (generalizes) **Requirement 84**
- **Evolution**: The class "**Evolution**" includes traceability relationships that describe the way in which a requirement is further developed (fulfilled, refined, replaced, extended, etc.).

  Example: **Requirement 73, Version 1.2** (replaces) **Requirement 73, Version 1.1**

It is not possible to give a general answer as to which relationship types/classes are relevant for a particular project. It is important to think about the goal and the traceability relationships to be used before starting the documentation (either as a company policy or project-specific) and to define these for all participants (see also [Maeder et al. 2009][Maeder et al. 2013]).

Example: To ensure that all requirements within the scope of a project are justified and that each requirement is tested, the following types of traceability relationships can be useful:

- Relationship of type "fulfilled" to ensure that there is no requirement that cannot be assigned to any business goal
- Relationship of type "tested" to ensure that a test case exists for each requirement
- Relationship of type " justified" to ensure that decisions about requirement changes have been documented

To ensure that traceability relationships are defined and used consciously, relevant artifacts and the relationship types between these artifacts should be documented in a traceability model (see EU 6.6). See also EU 2.3

EU 6.4 Forms of Presentation for Traceability Relationships (K1)

EU 6.4.1 Implicit and Explicit Documentation of Traceability (K1)

A traceability relationship can be documented either implicitly or explicitly. Although this learning unit focuses mainly on the explicit documentation of traceability, this section will address this distinction.

Explicit documentation of traceability: Explicit traceability is achieved through defined and deliberately established relationships between artifacts (cf. EU 6.4.3).

Implicit documentation of traceability: Implicit traceability can be achieved, for example, through naming conventions or documentation structure.

EU 6.4.2 Bidirectional and Unidirectional Traceability Relationships (K1)

Traceability relationships can be documented as unidirectional (directed) or bi-directional (not directed), depending on the goal to be achieved with traceability.

- **Unidirectional traceability**: allows traceability from one artifact to another, but not vice versa. For example, the reference from a test requirement to a system requirement allows checking why the test requirement exists or on what it depends. However, the system requirement will not be able to find a unique reference to a test requirement. This type of relationship is often found in document-based techniques, where relationships are maintained manually, for example by textual references, and refer to either the predecessor or successor artifact. In the documentation direction, it is important to note that reference is made to the artifact to which a dependency exists.
- **Bidirectional traceability**: allows traceability from one artifact to another and vice versa. Unlike the unidirectional relationship, it is possible to navigate between the artifacts, for example from a requirement to a test case (for example by a textual reference to a test case) and from a test case to the corresponding requirement that is to be checked with this test case. This type of relationship allows you to look at the predecessor and successor artifacts (Pre- and Post- Requirements Specification Traceability). In Requirements Management tools, these relationships are usually generated automatically, so that the tool supports navigation or impact analysis in both directions. For purely textual references, however, explicit maintenance is required for each artifact involved.

EU 6.4.3 Forms of Presentation for Traceability Relationships (K1)

For explicit documentation of traceability different forms of representation can be selected. See [Pohl 2010][Pohl & Rupp 2011]:

- **Text-based references**: The documentation and representation by textual references is the easiest way to implement traceability relationships between artifacts (see Figure 7).

The relationship describes the relationship type and a unique ID of the artifact to which the relationship refers (for example, [testcase_for → ReqID 1189]). This type of presentation has the advantage that it can be used independently of a Requirements Management tool and is easy to understand. It is usually documented directly in an artifact, e.g. in a test case there is a reference to a requirement.

Testfall	Referenz auf Anforderung	Testfallbeschreibung	Priorität
TC_0021	FR_0012 FR_0013 FR_0016	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea ...	Hoch
...			
TC_0150	FR_0020	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed ...	Hoch

Figure 7: Text-based references

- **Hyperlinks**: Unlike text-based references, hyperlinks allow direct navigation to the target artifact. Hyperlinks will be always created from the source artifact to the target artifact. Bidirectional relationships can be created by cross-referencing. Compared to simple textual references, using hyperlinks has the advantage that one can "jump" directly to the referenced artifacts. However, this is usually only possible within a tool.
- **Traceability matrices**: In a traceability matrix, traceability relationships are represented by references in the cells of a matrix (see Figure 8). The resulting matrix documents the relationship from the source artifact to the target artifact. This type of presentation allows an abstract presentation of dependencies between artifacts, e.g. BR_0010 is detailed by UC_10; BR_0020 is detailed by UC_30 and UC_40; UC_40 details (reverse reading direction) BR_0020 and BR_0030.

  Traceability matrices can be used to represent exactly one type of relationship between two artifacts. However, the presentation of different relationship types is also possible. Requirements Management tools such as DOORS automatically create these presentations based on artifacts and relationships, providing an overview of relationships between artifacts. In practice, however, such matrices quickly become very large and difficult to understand.

Source artifact \ Target artifact	UC_10	UC_20	UC_30	UC_40
BR_0010	is detailed by			
BR_0011		is detailed by		
BR_0020			is detailed by	is detailed by
BR_0030				is detailed by

Figure 8: Traceability matrix (BR = Business Requirement, UC = Use Case)

- **Traceability tables**: Unlike traceability matrices, traceability tables provide the ability to describe traceability relationships between all artifacts at different levels of detail (see Figure 9). They thus offer a powerful tool for documenting traceability from goals, through use cases and functional requirements, to test cases. This tool can be used independently of a specialized Requirements Management tool to document traceability among artifacts themselves documented in different tools (Word, Excel, Rational Rose, Visual Paradigm, Quality Center, etc.).

Business requirement	Use Case	Functional requirement	System requirement	GUI element	Test case
BR_0010	UC_10	FR_0012 FR_0013 FR_0016	CRM_0011 CRM_0020 DWH_0010 Billing_0020	GUI_0081	TC_0021 TC_0022 TC_0025
BR_0011	UC_20	FR_0020	CRM_0011 CRM_0020		TC_0060 TC_0150

Figure 9: Traceability table (BR = Business Requirement, UC = Use Case, FR = Functional Requirement, CRM = Customer Relationship Management, DWH = Data Warehouse, GUI = Graphical User Interface, TC = test case)

- **Traceability graphs**: In a traceability graph, the nodes represent artifacts and the edges represent the relationships between the artifacts (see Figure 10). To be able to distinguish between the different development artifacts (e.g. scenario, requirement, test case) and relationships (e.g. refines, implements, testcase_for) at a glance, it is recommended to define an appropriate notation form.

  However, their use is only recommended if these graphs can be created automatically based on the artifacts and relationships. A manual reproduction of such graphs and their maintenance will be too complex in practice. In principle, these graphs provide an easy-to-understand way of checking dependencies and navigating between the different artifacts.

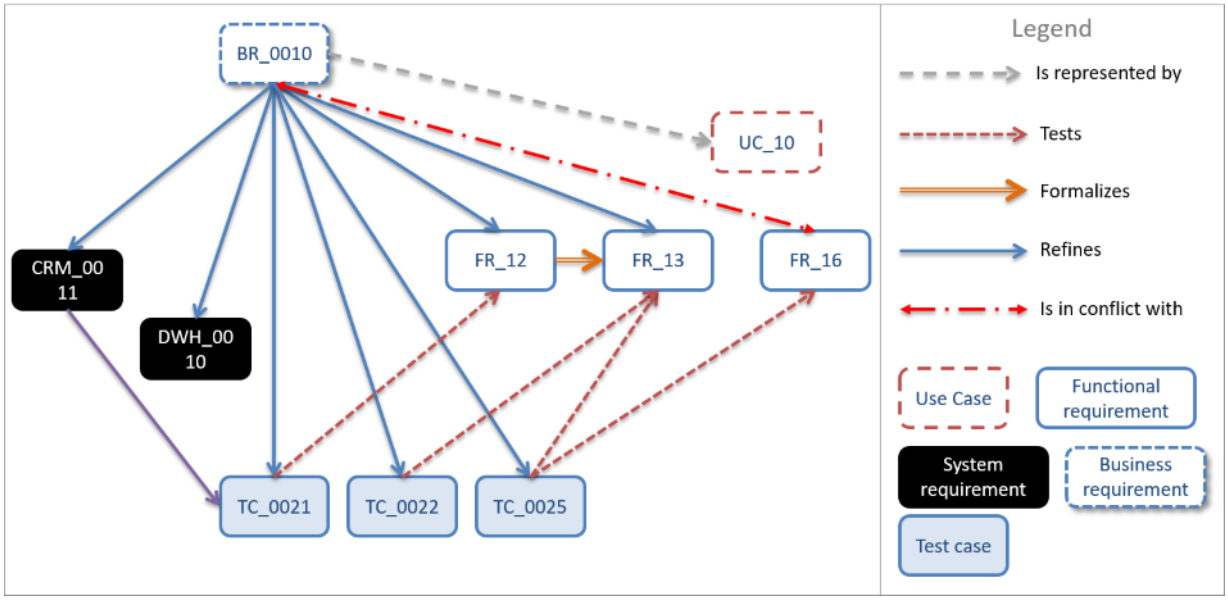


Figure 10: Traceability graph

Table 1 summarizes and evaluates the described forms of traceability relationships.

Form of presentation	Positive	Negative	Suitable for
Inline documentation of traceability			
Text-based references	Can be implemented independently of tools and comprehensively Relationship is visible in the artifact as plain text.	Traceability analyses are very complex.	To represent traceability in paper-based textual specifications.
Hyperlinks	Relationship is visible in the artifact as plain text. Easy navigation between artifacts to detect direct dependencies.	Traceability between different tools is not always possible. [Traceability between different tools not readily possible.]	To represent traceability in electronic specifications.
Orthogonal documentation of traceability			
Traceability matrices	Dependency between two artifacts is quickly and easily visible.	Manual creation of traceability matrices is time-consuming and leads to large, only poorly populated matrices.	Representation of a single relationship type between two specific artifact types (e.g. use cases and requirements)
Traceability tables	Enables clear display of the extended Pre- & Post-RS traceability. Allows a variety of traceability analyses.	High complexity of creation.	Representation of traceability between text-based and model-based artifacts in different documents / tools
Traceability graphs	Graphical presentation of traceability; allows "abstract" presentation of traceability relationships between artifacts.	Usage only possible with appropriate tool support.	Representation of complex traceability among artifacts in a Requirements Management tool.

Table 1: Forms of presentation of traceability relationships

EU 6.5 Development of a Strategy for Project-Specific Traceability (K2)

As already mentioned, establishing and using traceability in a project must be planned. It is not usually appropriate to document every relationship between artifacts. Instead, at the beginning of the project, one should think about why traceability is necessary in this project and at what points which kind of traceability will be required.

In addition to defining the relevant artifacts and relationship types, it is also necessary for a specific traceability strategy to define a system for recording and using this information.

A specific traceability strategy considers the following aspects:

- **Traceability goal**: Determining for what reason (i.e. why) traceability is necessary within the context of the respective project or what it is intended to achieve (i.e. what for) (cf. EU 6.1.2)
- **Usage strategy**: Definition of strategies for the use of traceability information by the development team. For example, a usage strategy could be change impact analysis, where traceability relationships are used to determine which artifacts are affected by a change.
- **Recording strategy**: Definition of strategies for recording of traceability information by the team. The responsibility for documentation of traceability relationships must be explicitly assigned in order for it to be carried out. The responsibility should be defined for each relationship type (for example, use case to functional requirement by the Business Analyst; functional requirement to test case by the Quality Manager). For example, one of the recording strategies could be the chronological documentation of traceability relationships proposed by [Hull et al. 2011] or [Wiegers & Beatty 2013]. The relationship between two artifacts is created as soon as the new artifact (e.g. requirement refinement or test case) is created. The advantage of this is that there is a clear responsibility for setting traceability relationships.
- **Project-specific traceability model**: Definition of the traceability information to be recorded and the form of presentation. A traceability model describes which types of relationships (e.g. is_refined_by; is_tested_by) between which artifacts (e.g. as a text-based reference of the other artifact on both artifacts) should be documented (see EU 6.6).

When defining a traceability strategy, attention shall be paid to the following:

- The entire team is aware of the need of requirements tracking,
- the traceability model is understandable and accepted by all parties involved,
- team members know and accept the responsibilities assigned to them for the documentation of traceability relationships,
- the necessary prerequisites are established to correctly document traceability.

EU 6.6 Creating and Using Specific Traceability Models (K2)

In order to create a project-specific traceability model, one should first consider among which artifacts traceability should be established and which traceability relationships between these artifacts are necessary (allowed). These specifications should be described by a project-specific traceability model (cf.[Maeder et al. 2013] and Pohl 1996]) and communicated within the project.

In a specific traceability model all project participants can clearly see which artifacts exist, which relationship types are to be maintained and who has to maintain them and how (cf.[Pohl 1996][Pohl 2010][Maeder et al. 2009]).

EU 6.6.1A Process for Defining a Specific Traceability Model

The following describes a sample process for defining a specific traceability model.

1) Selection of a referencing schema

   The first step should be to check whether an existing traceability model can be reused and adapted. An effective way to define a specific traceability model is to reuse an existing traceability model from a similar project or a company-wide traceability model. Such a traceability model can serve as a basis for defining the specific traceability model and will usually already contain a large number of artifacts and dependencies to be defined.

2) Selection of artifacts

   In this step, it is determined between which artifacts traceability should be ensured in order to support the goal set in the traceability strategy and the usage scenarios, e.g. traceability between use case and functional requirement and between requirement and test case.

3) Definition of permitted relationship types between types of artifacts

   Here it should be specified which relationship types are allowed for representing traceability (see EU 6.3) between two artifact types, e.g. a valid relationship between requirement and test case is: "validated by".

4) Determination of the number of traceability relationships (at instance level)

   Here the minimum number of relationships between the real artifacts is specified (at instance level of the traceability model), e.g. each requirement requires one traceability relationship to a test case.

5) Definition of the dependency between artifacts

   Here it is specified which artifact is dependent on another artifact, e.g. a test case depends on the content of a requirement. When using unidirectional relationships, attention should be paid to referencing (see EU 6.4.2)

EU 6.6.2Using a specific traceability model

In addition to the definition of artifacts and traceability relationships, e.g. documented in an information model, further aspects have to be considered for the implementation and use of a specific traceability model:

1) Definition of the form of presentation

   After defining which relationships between which artifacts should be documented, it must be clarified in which type and form of presentation traceability relationships should be documented. The selection of the form of presentation for traceability relationships is usually influenced by the form of representation of the artifacts (see EU 6.4.3).

2) Providing support for recording data

   Recording traceability relationships between artifacts represents an additional effort (see 0), which usually serves other stakeholders (e.g. project managers). Therefore it is very helpful if the documentation of traceability relationships is supported as far as possible. This can be supported on the one hand by Requirements Management tools - or by self-programmed solutions for example with Word macros.

3) Aligning a tool with project artifacts

   When using an RM tool, a translation into the existing terminology of the tool is usually required. In this step, identifiers of artifacts and relationship types defined in the model are linked to identifiers offered by the tool and referenced uniquely. For example, if the tool only offers one artifact type "Requirement", but the traceability model distinguishes between "User Requirement" and "System Requirement", then an appropriate mapping and, if necessary, assignment of an additional attribute is needed here, allowing later differentiation.

EU 6.7 Measures for the Evaluation of Implemented Traceability (K1)

Traceability inevitably raises the question of how well and completely traceability information among artifacts (requirements, decisions, fragments of code, test cases, etc.) is actually documented and whether the implemented traceability fulfils its actual goal (see EU 6.1).

Checking traceability information provides an insight into the quality of the current documentation. Furthermore, these results are also helpful in identifying deadlocked processes or "unsuitable" traceability models.

The following sample measures can help to check the completeness and quality of traceability relationships:

- Ratio of the number of correct traceability relationships to the total number of traceability relationships (correctness).
- Ratio of the number of existing traceability relationships to the total number of traceability relationships required (completeness).
- Ratio of the number of requirements with traceability relationships to the total number of requirements (density).

An insufficient proportion of relationships between artifacts suggests that the relationships have not been maintained consistently. On the other hand, a low ratio of correct relationships suggests that either relationships were negligently maintained or that changes were not consistently applied to all the artifacts concerned. Any deviation from the stated goal may have different reasons that need to be discussed. The corresponding threshold values, for which actions must be taken, should be defined specifically.

In addition to recognising that the defined traceability strategy is not implemented, the question arises as to why traceability is not implemented or is only implemented incorrectly.

Possible reasons for missing or incorrect documentation of traceability are:

- Necessity of traceability is not recognized.
- Lack of traceability strategy (who documents what and why).
- Time constraints do not allow documentation of traceability.
- There is no agreed traceability model.
- Insufficient tool support when recording traceability relationships.

EU 6.8 Challenges in the Traceability of Non-textual Artifacts (K1)

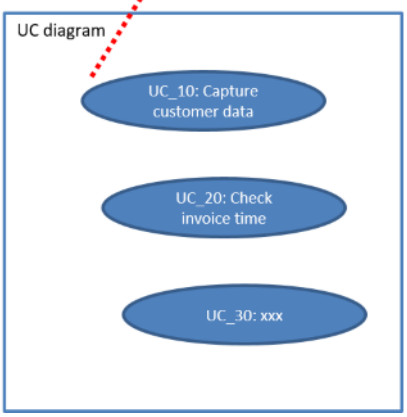
Traceability between textual artifacts (e.g. functional requirements) and model-based artifacts (e.g. activities in UML activity diagrams), or among model-based artifacts themselves can only be achieved with high effort.

Regardless of how the traceability relationships were created, the use of traceability information for evaluations and analyses (e.g. what effect does changing a business requirement have on the existing software) is still complex, since it is usually not possible to open the corresponding artifact directly (as with a hyperlink). In the end, however, it is this maintenance that makes an impact analysis possible at all.

One of the tool-independent options for documenting references between textual artifacts and model elements is, for example, traceability tables (see EU 6.4.3), which can clearly reference both the textual artifacts and the model elements. To do this, however, it is necessary to assign unique IDs (manually or automatically) to the elements in the model. Using a textual reference or a traceability table it is possible, for example, to clearly reference model elements such as use cases to corresponding textual requirements.

Figure 11 shows a simple example of referencing textual requirements and use cases in a use case diagram using a traceability table.

Business requirement	Use Case	Functional requirement	System requirement	GUI element	Test case
BR_0010	UC_10	FR_0012 FR_0013 FR_0016	CRM_0011 CRM_0020 DWH_0010 Billing_0020	GUI_0081	TC_0021 TC_0022 TC_0025
BR_0011	UC_20	FR_0020	CRM_0011 CRM_0020		TC_0060 TC_0150



FR_ID	Functional requirement	Priority
FR_0012	The name of the customer must be checked against the inventory customer data before the entry.	high
FR_0013	The bank account of the customer must be validated against the SEPA HUB	high
...	...	...

Figure 11: Traceability among textual and model-based artifacts

EU 7 Variant Management for Requirements (K2)

Duration: 3 hours
Terms: Variability, Reuse, Product Family, Product Line, Feature, Feature Model, Variation Point, Variant, Binding Time

Educational Objectives:
EO 7.1.1 Knowing the reasons for using variants of requirements (K1)
EO 7.1.2 Knowing key terms in the field of "Variant Management for Requirements" (K1)
EO 7.1.3 Knowing the benefits of explicit documentation of variability (K1)
EO 7.2.1 Knowing common forms of presentation for creating variants of requirements (K1)
EO 7.2.2 Analyzing a given form of presentation of variability according to given criteria (K2)
EO 7.2.3 Evaluating a given form of representation of variability with respect to an operational situation in a given context (K2)
EO 7.3.1 Knowing the concept of feature modeling (K1)
EO 7.3.2 Mastering and using a feature model (K2)

EU 7.1 Use of Variants of Requirements (K1)

In many cases, product development involves not only creating a single product, but also - either in parallel or sequentially - a series of similar products. In this case it usually makes sense not to run the product developments independently of each other, but to reuse development artifacts such as requirements, architectures, program code or test cases appropriately [ISO 26550].

If similar products are developed by the systematic and planned reuse of development artifacts, it is referred to as a product line or product family [Clements & Northrop 2007].

The definition of the term product line is as follows: *"A **software product line** is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way."* (from [Clements & Northrop 2007]).

Core assets typically include requirements. Requirements for a set of products are not managed independently of one another; instead, requirements are appropriately located in a common "requirements pool" and are assigned to the individual products.

The product line concept can also be limited to the level of requirements only. Also in this case, requirements for different products are managed in a "requirements pool" and requirements for special products are then derived from this pool, even if the product development of the different products then proceeds independently. In this case it is referred to as a requirements product line.

In both cases, a requirement pool is a set of requirements that contains more than the set of requirements for a specific product. A requirement pool can also contain requirements that are not currently included in any product.

In product line development it is common to distinguish between two processes [Weiss & Lai 1999][ISO 26550]:

- Domain engineering: in domain engineering a "basic product" will be developed
- Application engineering: here the "basic product" will be adjusted to the specific product

A central concept for reusing requirements is **variability**. [Pohl 2010], [Pohl et al. 2005]. Variability allows the definition and realization of different products by selecting concrete variants from a given set of possible variants.

Each variant always refers to a variation point. Variation points describe where something varies within a product line, variants describe possible (permissible) specifications (of requirements) at a variation point. Usually, a requirements document contains a large number of variation points with typically several variants. However, not every combination of individual variants is permissible. There are typically variant dependencies. Variant dependencies can be optional or mandatory. Optional means that a variant can be selected, but does not have to be. Mandatory means that a variant must be selected (e.g. from several alternatives).

By selecting specific variants for each variation point, a specific product is "configured". If all variant dependencies are considered, the result is a valid, specific product.

In practice, not all variation points are assigned with concrete variants during requirements collection, i.e. one of several possible variants is selected. Some variation points remain open, i.e. they are not yet "bound" to a specific variant. The binding time (cf. [Coplien et al. 1998]) can remain open until the system is delivered or until system operation.

Possible binding times are "before development", "during development" (implementation time), "during assembly" (build time), "during commissioning" (installation time), "during system launch" (startup time) and "during runtime" (runtime) [Atkinson 2002].

Variability can be documented implicitly or explicitly. In the case of implicit documentation, the reader must recognize from the formulation of a requirement that different product variants (product specifications) are possible here.

In **implicit** documentation, the word "or" often indicates that different product versions are possible. However, the word "or" is not a reliable indicator of a variation point, as it is also very often used in logical conditions. In addition, other key terms are also possible, which likewise indicate different product specifications (e.g. "both... as well as"). These are usually not clear enough either.

The **explicit** documentation of variability can be integrated into requirements documentation or orthogonally, i.e. in a separate model. In the case of textual requirements, both the variation points and the possible variants are explicitly shown in the requirement text in an integrated documentation.

In the case of orthogonal documentation, the textual requirement remains untouched. Documentation of variation points and variants is done in a separate model.

In product line development in general and in Requirements Management in particular, explicit documentation of variability has the following advantages [Pohl 2010]:

- **Communication**: The explicit documentation of variation points and possible variants supports communication with the affected stakeholders, as it is easy to see which variants can be selected at which points.
- **Decision support**: The explicit documentation leads to more conscious decisions, (1) at which points variability is provided and (2) which concrete variant was selected for a given product.
- **Traceability**: By explicit documentation of variant dependencies, dependencies can be analyzed and, in case of requirement changes, used to analyze subsequent changes.

EU 7.2 Forms of Explicit Documentation of Variants and their Evaluation (K2)

In practice, many different forms of documentation of variants can be found in requirements documents. These forms use the concepts introduced in section EU 7.1 such as variation point, variant, product allocation and documentation of binding times in very different ways.

First, common forms of representation [Boutkova 2011] will be presented in sketched form and illustrated with a short example from the automotive industry. Then these forms of presentation will be analysed with regard to the concepts introduced in EU 7.1. Subsequently, criteria for evaluating strengths and weaknesses of the forms of presentation will be identified and the presented forms evaluated.

Feature models represent another form of representation. These are discussed in more detail in section EU 7.3.

Form 1: Textual assignment of requirements to concrete products

In this case, the affected products are explicitly named in the individual requirements (Figure 12).

ID	Requirement
R32	The A-Class sun visor should be plastic-coated.
R33	The E-Class sun visor should be covered with leather.
R34	The sun visor in all products should contain an illuminated make-up mirror.

Figure 12: Textual assignment of requirements to concrete products

Form 2: Explicit assignment of requirements to specific products

In this case, the individual requirements are assigned directly to the products (product variants) concerned (Figure 13).

ID	Requirement	A-Class	E-Class
R32	The sun visor should be plastic-coated.	X	
R33	The sun visor should be covered with leather.		X
R34	The sun visor should contain an illuminated make-up mirror.	X	X

Figure 13: Explicit assignment of requirements to specific products

There are various sub-variants related to the concrete form of direct assignment:

- Explicit assignment to separate product columns
- Multiple selection in a product column (see also first product column in Figure 14)

Form 3: Explicit assignment of requirements to specific product features

In this case, individual requirements are assigned directly to several product features (Figure 14). A specific product is defined by several product features, which can have different characteristics. A requirement belongs to a product if the requirement is assigned to the product features belonging to the product. For example, requirements R33 and R34 belong to the "E-Class in the USA" product.

ID	Requirement	Series	Market
R32	The sun visor should be plastic-coated.	A-Class	USA Europe
R33	The sun visor should be covered with leather.	E-Class	USA
R34	The sun visor should contain an illuminated make-up mirror.	A-Class E-Class	USA Europe

Figure 14: Explicit assignment of requirements to specific product features

Note: This form of presentation bears the risk of invalid configurations suddenly becoming valid due to combinatorics. Example: In case of R34, the requirement shall only apply to the A-Class USA and the E-Class USA and E-Class Europe. Exclusion of A-Class Europe cannot be represented by the chosen classification methodology.

Form 4: Indirect assignment of requirements to products through features

In this case, the individual requirements are assigned to product features. A separate product configuration then determines which product features are contained in a specific product (that is, a specific product variant), which indirectly determines the relevant requirements (Figure 15).

ID	Requirement	Feature
R32	The sun visor should be plastic-coated.	Plastic surface
R33	The sun visor should be covered with leather.	Leather surface
R34	The sun visor should contain an illuminated make-up mirror.	

Product	Features
A-Class	Plastic surface and ...
E-Class	Leather surface and ...

Figure 15: Indirect assignment of requirements to features and product configuration

Requirements not assigned to any feature apply to all products.

Analysis of the forms of presentation

The following aspects are to be considered in the analysis of forms of presentation, according to EU 7.1:

- **Description of binding times**

  All the forms of presentation described are limited to one binding time only. In this case, only the binding time "during development" (implementation time) is considered.

- **Variation Points and Variants**

  Variation points can be identified only indirectly in these forms of representation in that there are several requirements that are obviously contradictory and the contradiction can only be resolved by assigning variants to specific products (see requirements R32 and R33). The specific variants are described directly in the requirement text.

- **Variant Dependencies and Verification**

  In the presented cases, variant dependencies are not documented (forms 1, 2, 3) or only indirectly documented (form 4) and therefore cannot be verified. Errors in the configuration in the best case can be identified by a human inspector if, for example, conflicting requirements are selected for the same product (if, for example, R32 and R33 had been selected for the A-Class). In Form 4, a variant dependency can be deduced indirectly if there are several requirements that refer to the same features.

Analysis of Strengths and Weaknesses

When evaluating a specific form of presentation used for variability, the following criteria are relevant for practical application [Boutkova 2011]:

- Teachability: How easily can the chosen form of presentation be trained to non-technical personnel?
- Scalability: How easily can the chosen form of presentation be used for a large number of products?
- Expandability: How much effort is necessary to configure a new product?

- Migratability: To what extent can existing requirements documentation be further developed in the direction of the chosen form of presentation without explicit variability information?
- Verifiability: To what extent can incorrect configurations in the selected form of presentation be automatically identified?
- Comparability: To what extent can requirements of different products be easily compared?
- Changeability: How easily can existing requirements for a single product be changed without affecting other products in the product family?

EU 7.3 Feature Modeling (K2)

Feature modeling is a common technique for documenting variability. The best known representative of feature modeling is FODA - Feature-Oriented Domain Analysis [Kang et al. 1990]. Feature modeling has in the meantime become widely used, especially in product line development, and there are many extensions and additions to the original FODA approach[Schobbens et al. 2006].

A feature is thereby defined as a „*prominent or distinctive user-visible aspect, quality, or characteristic of a software system or system*" [Kang et al. 1990]. A feature model describes features and their interdependencies. A product variant consists of a set of features that describe the product.

A (valid) product configuration and the features it contains are defined by the boundary conditions specified by the feature model. Feature models are often represented graphically in the form of a feature diagram.

The descriptive elements of a feature model can be divided into the following three categories:

- Basic elements
- Advanced elements
- Cardinality-based elements

The basic elements of a feature model describe parent features and their children and express the relationships among them. Child features can have the following relationships with parent features:

- Mandatory - The child feature is mandatory
- Optional - The child feature can be used
- Or - At least one of the child features must be selected
- Alternative - Exactly one of the child features must be selected.

The advanced elements can be used to define additional dependencies between features. The best known dependencies are

- A requires B - The selection of feature A implies the selection of feature B.
- A excludes B - Features A and B cannot be contained in the same product.

Cardinality-based elements can be used to further specify the allowed relationships between basic elements, for example by adding notations such as [min, max] to the parent-child relationship.

The notation used below is taken from [Czarnecki & Eisenecker 2000].

Figure 16 shows a sample feature model. Feature F contains two mandatory features f1 and f4, where f1 contains two mandatory features f2 and f3, while f4 contains only one optional feature f5.

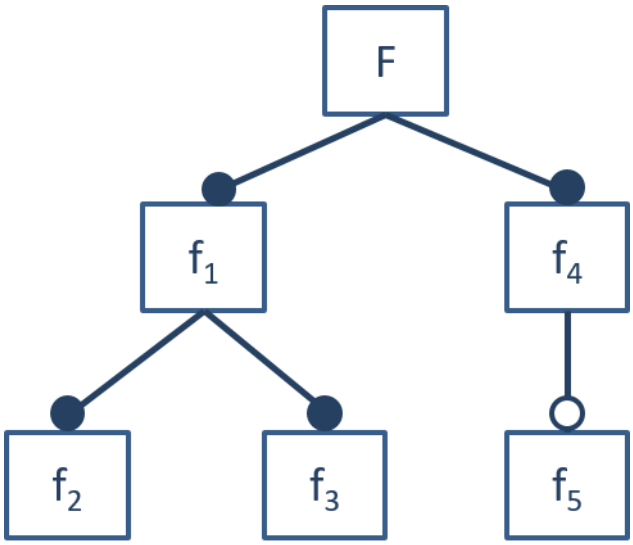
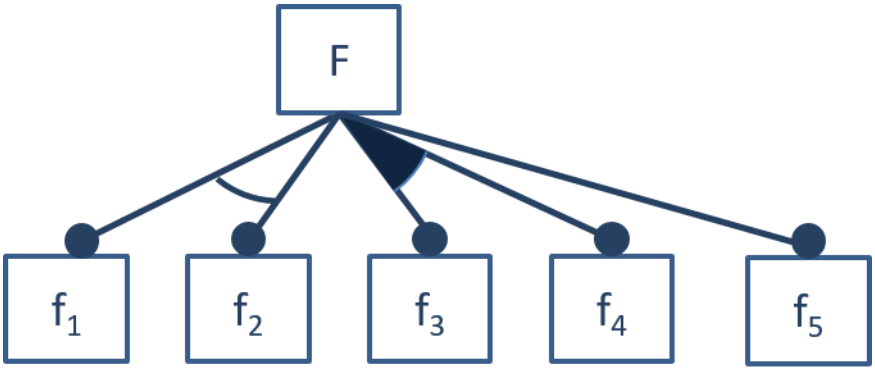


Figure 16: Example of a simple feature model

Figure 17 shows the presentation of alternative- (empty arc) and or-relationships (filled arc) in feature models. On the right side all possible product configurations resulting from this model are shown.



Possible product configurations:
{F; f1; f3; f5}, {F; f1; f3; f4; f5} {F; f1; f4; f5}
{F; f2; f3; f5}, {F; f2; f3; f4; f5} {F; f2; f4; f5}

Figure 17: Representation of alternative- and or- relationships (left) and resulting possible product configurations (right)

An additional example of dependencies between features can now be found in Figure 18.

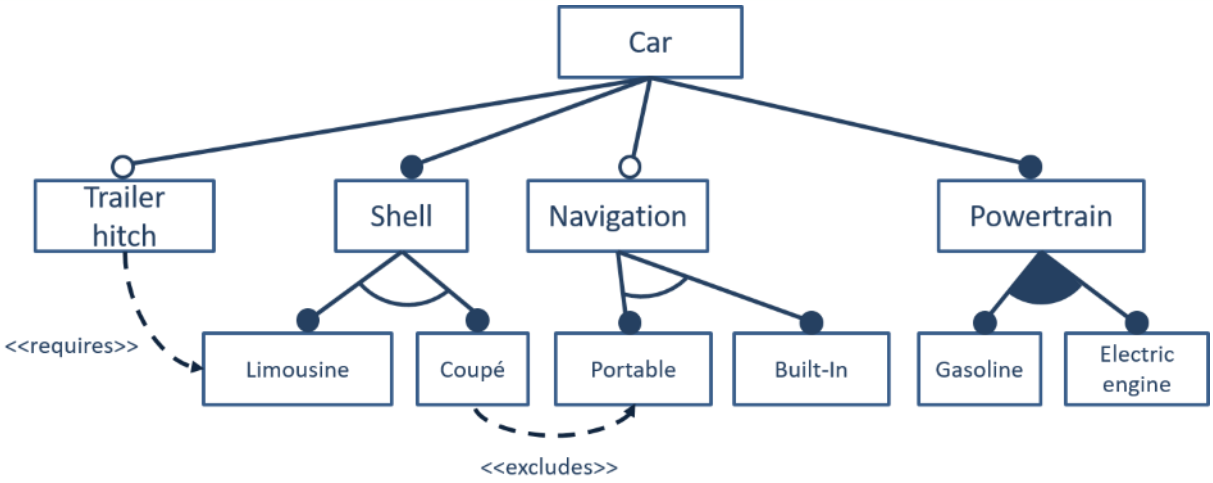


Figure 18: Feature model for a car with dependencies

In this example, a trailer hitch can only be fitted if it is a limousine.

A coupé excludes the installation of a portable navigation device. The engine can be a purely internal combustion engine (petrol), purely electric (electric motor) or a combination (i.e. hybrid drive).

In feature modeling, variants are represented by leaf elements in the feature model. Variation points are non-leaf elements.

The group-refining relationships "Or" and "Alternative" have a higher value than the single-refining relationships "Mandatory" and "Optional".

When using "Or" and "Alternative", the use of the "Optional" or "Mandatory" labels for the affected child features can therefore be omitted.

An example of cardinality-based elements can be found in Figure Figure 19. Product F has to have two or three of the features f_1 to f_4.

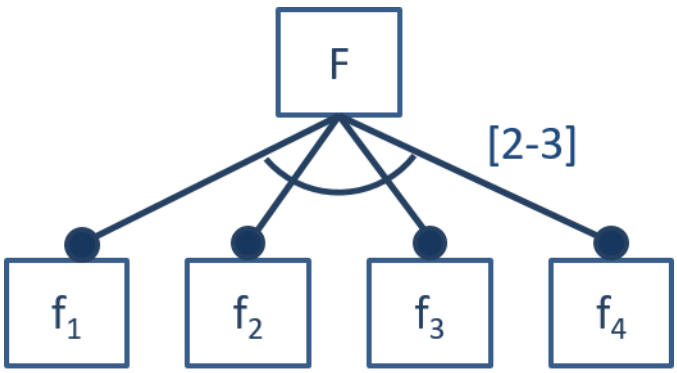


Figure 19: Feature model with cardinality-based elements

Identification of Features

If feature-based Variant Management is to be introduced, the question arises as to which features will be used in the future. As a rule, it makes sense not to invent completely new features, but to use existing requirements documents when defining features [Boutkova & Houdek 2011].

A good starting point for identifying features is considering the nouns in a requirement text as potential features. It is important to ignore general nouns that have nothing to do with the product as such (e.g. in sections dealing with contractual aspects or the development process such nouns are also called stop words). Based on such noun lists, an expert can then usually quickly identify potential features. However, one disadvantage of this procedure is that particularly variation points that are not explicitly mentioned in the text cannot be identified.

Variation points can often be identified when the Requirements Analyst persistently asks for the "why" for different variants.

Tool Support

If variability should be explicitly documented, this is usually not possible without the use of special tools. The are tools available on the market which allow the user to

- Create feature models,
- Build product configurations,
- Check the admissibility of product configurations.

Typically, such tools have distinct interfaces to other tools in which the actual development artifacts (for example, requirements or test cases) are located and which thus make it possible to relate the development artifacts to the characteristics.

EU 8 Reporting in Requirements Management (K2)

Duration: 1,75 hours
Terms: Reporting, Report, Key Figure [Measure], Goal-Question-Metric-Method

Educational Objectives:
EO 8.1 Knowing the goals and benefits of reporting in RM (K1)
EO 8.2 Knowing the interfaces, contents and the definition process for a report (K1)
EO 8.3.1 Knowing typical key figures in Requirements Management (K1)
EO 8.3.2 Mastering Requirements Management key figures using the Goal-Question-Metric method (K2)
EO 8.4 Knowing risks and problems in the use of reporting (K1)

EU 8.1 Goals and Benefits of Reporting in Requirements Management (K1)

Reports are part of project and organizational controlling. They serve to collect information about projects or organizational units and to prepare it appropriately for certain target groups in order to meet their information needs.

[Ziegbein 1998] defines reporting as "the creation and dissemination of cross-functional reports in the sense of an organized compilation of messages exclusively for management". Another definition emphasizes the preparation and goals of the reporting system: "It can be understood as all persons, facilities, regulations, data and processes used to create and distribute reports. Thereby reports represent summarized information under an overarching goal, an information purpose.". [Küpper 2005]

Reporting in Requirements Management (RM) is the collection, evaluation and presentation of information about requirements or the Requirements Management process and the provision of this information.

A report is a document that combines one or more views for a specific stakeholder and purpose.

The following comments on reporting in Requirements Management focus primarily on the development of new systems. However, these procedures and principles can also be applied in other contexts such as the continuous enhancement of systems as well as in the context of cross-project management of requirements.

EU 8.2 Establishment of a Reporting System in RM (K1)

EU 8.2.1 Interfaces (K1)

Requirements Management is closely integrated with Project Management, Product Management and Quality Management. Consequently, these interfaces also exist within the reporting system. Therefore it makes sense to coordinate the reporting of these three areas and their data. Project Management and Quality Management will be the most important recipients of reporting on Requirements Management.

Reports from Requirements Management and Quality Management are often created collectively.

EU 8.2.2 Contents of a Report (K1)

In principle, reports can be created in any form (e.g. informally in an e-mail text). However, there are many templates, so every report has the same structure. This makes reports easy and efficient to read and create. The same information is always in the same place in the report. For the author, it is particularly practical if the report can be generated automatically from the tool in which the necessary information is managed anyway.

The standard contents of reports are the following:

- **Project name:** The report must specify to which project it refers. (If the report is for an organizational unit, for example a department, the name of the department is displayed here instead of a project name.)
- **Date of report creation:** The contents presented in the report change daily or even hourly. It is therefore important to specify when the data was extracted: that is, on what information state the report is based.
- **Version number:** If there are several versions of a report, for example because someone made an addition, the new version must have a new version number to ensue the uniqueness of the report and better traceability of changes.
- **Reporting period:** Reports can refer to days, weeks, months, years or any other time interval. Weekly and monthly reports are the most common, but in critical project phases reports can also be generated on a daily or half-day basis. Of course, when interpreting the contents of the report, it makes a difference whether it relates to what was achieved within a week or a month.
- **Creator and recipient:** A report has a creator (author) and recipient (distribution list). The recipients can also be distinguished between those who receive it for information only and those who have to approve it. The names of these persons are usually mentioned on the report and are thus documented.
- **Release status**: If the report requires a release, this status should be noted here. The report may contain different contents in different release statuses.
- **Overall status**: Right at the beginning of the report, the reader in a hurry would like to get an overview of how critical the project is. Busy managers only read the report if the project is critical. Reports on projects that go according to plan do not contain any informational value for the supervisor, as his or her support is not required. Traffic light scales with the well-known colours green, yellow and red are popular.
- **Technical content:** This is the core of the report.

The technical content of a Requirements Management report consists of one or more of the following content classes:

- Selective or summarizing **views on requirements** (see EU 3.5). These can usually be generated automatically by a Requirements Management tool.
- **Key figures** about requirements or the RE process. These key figures are usually obtained by means of automated evaluations. This evaluation can be done with a Requirements Management tool or outside.
- Descriptive or evaluative **descriptions**.

EU 8.2.3Tips for the Development and Application of Reporting (K1)

In the development and application of (requirements-based) reporting, there are some practical tips or hints that need to be considered:

- **Focusing on the essentials:** Even when the stakeholders and the benefits of reporting are known, the art is to focus on the essentials. The GQM method, which we describe in chapter EU 8.3.2, helps here.
- **Reconciliation:** The information required for the report must be provided in a requirements information model and attribute schema. As it is difficult to subsequently change the information model and attribute schema and the introduction of a new attribute requires extensive content maintenance, the RM data models should be clarified early, even before a potential development project. It is helpful to use reference models that have already been coordinated with each other.
- **Data collection:** Those who have to collect the data are not the same ones who need the information and create or read the report. The data collectors therefore have no inherent motivation to enter the data. It is therefore even more important that data collection is well integrated into daily work processes and that it is clear who has to enter which data and when.
- **Data quality:** The mere presence of attributes does not necessarily mean that all content is maintained, up-to-date and correct. While it does not make sense for an efficient work process to introduce too many mandatory fields, especially since some information is not yet available when creating a requirement, for reporting it would be important that the attributes are maintained. Missing content leads to incomplete information in the reports.

EU 8.2.4Report Definition Process (K1)

The Requirements Manager is responsible for defining the requirements-based reports and coordinating them with the Requirements Information Model, or delegates this task to an appropriate person.

According to ISO 15288 ([ISO 15288], 6.3.7.3 a) 1) to 4)), a measurement and reporting system is defined in these steps:

1) Description of characteristics of the organisation relevant to the measurement.

2) Identification and prioritization of information needs.
3) Selection and documentation of key figures that meet these information needs.
4) Definition of procedures for data collection, analysis and reporting.

EU 8.3 Key Figures in Requirements Engineering (K2)

Key figures or measures (often wrongly referred to as metrics) are an important part of reports. [Ebert 2012, p. 436] defines a measure as:

"(1) A formal, precise, reproducible, objective assignment of a number or symbol to an object to characterize a specific characteristic.

(2) Mathematical: Figure M of an empirical system C and its relations R in a numerical system M.

(3) The use (collection, analysis, evaluation) of a measure. Examples: Measure for a product (for example defect, duration, deviation from plan) or a process (for example defect costs, efficiency, effectiveness)".

EU 8.3.1 Key Figures in Requirements Management (K1)

Key figures in Requirements Management can be divided into two major classes:

- **Product key figures** (i.e. key figures on requirements):
  These include key figures on the number of requirements and the characteristics of requirements.
  Examples: Number of safety-critical requirements, average length of a requirement, number of faulty requirements,
- **Process key figures** (i.e.key figures on the RE process):
  Examples: Requirement changes per month, average effort to check a requirement, average frequency of changes to requirements in the last n months

EU 8.3.2 Deriving Key Figures Using the Goal-Question-Metric Method (K2)

"Goal-Question-Metric" (GQM) [Basili 1984] is a potential method for ensuring that only goal-oriented key figures are defined for reports or report content. GQM is a systematic procedure for identifying such key figures. A suitable key figure is identified by answering the following questions:

- Which goal is to be achieved by the measurement? (*Goal*)
- What should be measured and which questions should the measurement answer? (*Question*)
- Which key figure(s) can describe the necessary characteristics? (*Metric*)

EU 8.4 Risks and Problems in Reporting (K1)

In practice, there are practical difficulties in gathering and evaluating data that result in reports not adequately reflecting reality. As reports are intended to lead to important management decisions, an incomplete or even deliberately embellished report can have far-reaching consequences.

Evaluation of data: condensed representation of reality

A report is always a highly condensed model of reality in which similar things are grouped into categories and insignificant details are omitted. It is very difficult to do this in such a way that any future question can be answered well at any time.

That is why the superficiality of a report must always be taken into account. In particular, it is important to avoid drawing false conclusions from the available data. For example, a report that shows 99% traceability for all project requirements does not yet allow a statement on the progress of the project or the quality of the relationships. Requirements that have not yet been linked could be the most important or most time-consuming requirements that contribute significantly to the success of a project. When reducing the complexity of key figures, one should always be aware of this problem. Often only very rough statements and conclusions are possible.

Poor Data Quality

Missing data is usually easy to detect. It is not as easy to evaluate the quality of the data: Do the data correspond to reality? Are they up to date? Do they measure exactly what they should, for example, does the attribute "effort" only measure the implementation effort, although the test effort should also be taken into account? Is the criticality really the result of an expert survey or has it been set provisionally?

Undiscovered but also known shortcomings in data quality lead to the fact that the report does not correctly reflect reality in new and further developments. Due to wrong data, it is difficult to make the right management decisions. And even if the lack of data quality is known, decisions are difficult to make.

Poor data quality often results from the fact that the parties involved neglect data maintenance because they themselves have little benefit from it. Conversely, sometimes they may be interested in making the data too nice, or at least in saving time on data maintenance by not carrying out careful analyses, but entering data that seems hastily plausible.

However, poor data quality can also result from the fact that not everyone involved has the same vision. In agile development (cf. EU 10) the "*definition of done*" is an important topic of discussion. It must be clearly defined when a requirement is considered completed. Possible criteria for the implementation of a requirement are, for example: the code has been created, unit tests have been created and successfully run, the documentation has been adapted and the code convention followed.

Data Protection Regulations

Applicable general and company-specific Data Protection Regulations must be followed when defining and implementing the reporting system. If personal data is provided by participants and further communicated within the company in the form of reports without their knowledge, this can lead to problems. In this context, it is important to clearly agree with the data creators who receives which data within the scope of the decisions to be made. In general, personal and person-related data should be used sparingly, or not entered in the first place. When defining views, care should also be taken to ensure that no statements about individual persons can be made so as not to unintentionally violate data protection regulations.

Inflationary Reporting

If the volume of report information increases constantly, this might also lead to a situation where the report recipients are unable to process this data due to time constraints and important decisions can no longer be made on a sound basis.

Therefore less is more! Focusing on the really necessary information is to be aimed for. This can also mean that different target groups receive different reports in which only certain aspects are represented, or at various levels of detail.

EU 9 Management of Requirements Engineering Processes (K2)

Duration: 2 ½ hours
Terms: iterative RE, upfront, lightweight requirements, Requirements Engineering Process (RE process), PDCA (Plan Do Check Act) cycle, Continuous Process Improvement (CPV)

Educational Objectives:
EO 9.1 Knowing Requirements Engineering as a process (K1)
EO 9.2.1 Knowing selectable parameters of the RE process (K1)
EO 9.2.2 Assessing the suitability of an RE process with regard to process parameters (K2)
EO 9.3 Knowing and applying different methods for documenting the RE process (K2)
EO 9.4 Knowing the necessity of monitoring and controlling the RE process (K1)
EO 9.5.1 Mastering and using methods to improve the RE process (K2)
EO 9.5.2 Knowing the necessity of a Requirements Management Plan RMP (K1)

EU 9.1 Requirements Engineering as a Process (K1)

A process consists of interdependent activities performed to achieve a specific goal. For each activity, inputs (information, material, energy, resources) are transformed into outputs (results) [ISO 9000]. Each activity is uniquely assigned to a responsible organizational entity, for example, a role. Thus Requirements Engineering and Requirements Management are also a process.

The RE process is a process for collecting and managing requirements: „A systematic process of developing requirements through an iterative co-operative process of analyzing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained." [Loucopoulos & Karakostas 1995, S. 13]

This RE process includes the following main activities [IREB FL 2012, EU 1]:

- Eliciting requirements
- Documenting of requirements
- Requirements validation and negotiation
- Requirements management

In each specific project there are several investigative activities such as workshops and meetings with stakeholders, document analysis and so on. The same also applies to the other main activities.

The RE process uses stakeholders' needs and ideas as input information. In addition, the status quo before project start (e.g. the legacy system) and competing products also play a role. The result of the RE process is a validated, conflict-free, consistent, prioritized, quality-assured requirements specification that can serve as a reliable basis for further project work.

In general, the four main activities have the following input information and results, which can of course look different, especially if company-specific requirements or standards have to be met (see Table 2):

Main activity	Input	Result
Eliciting Requirements	Stakeholders and their needs and ideas; If applicable: an existing legacy system and its documentation; competitor products	Oral and written requirements including the system vision
Documenting of requirements	Oral and written requirements	Written requirements specification (textual or model-based or both)
Requirements Validation and Negotiation	Written requirements specification	Validated, conflict-free, consistent, prioritized, quality-assured requirements specification
Requirements Management	Written requirements specification and change requests	Always up-to-date, validated, conflict-free, consistent, prioritized, quality-assured requirements specification; Preparation of requirements for individual stakeholder groups

Table 2: Four main Requirements Engineering activities and their inputs and results

These main activities must always be performed, whether explicitly or implicitly documented. Different standards require different implementation of these activities and set different guidelines regarding artifacts (see also EU 1.4).

The results of the RE process must meet quality criteria in three independent dimensions: specification, representation and agreement [Pohl, 1994]. Requirements should become more mature over time within these dimensions.

- **Specification:** This dimension describes the completeness of the specification. At the beginning of the RE process, requirements are vague and unclear (opaque). As the process progresses, requirements become more complete in the sense of a thorough coverage of the problem to be solved and a description that is detailed enough to be properly understood.

  Various standards provide guidelines as to which conditions must be met by the requirements in order for them to be considered complete. However, it is not possible to prove the completeness of requirements.
- **Presentation:** The scale varies from informal to formal. Informal presentation includes sketches, free text and prototypes. Semi-formal presentation includes graphical models such as class diagrams, state machines, use case diagrams or data flow diagrams. Tabularly presented use cases, which strictly follow a given syntactic structure, are also semi-formal. Formal specifications clearly describe requirements using logic languages and formal semantics. Preparation of a formal specification usually begins with informal forms of presentation.
- **Consent:** Establishing agreement is another goal during the RE process. The agreement dimension moves from the personal view to a common view of the requirements.

Requirements specification should be optimized in all three dimensions. Here elicitation activities mainly contribute to improvement in the specification dimension, documentation activities to the presentation and validation and negotiation activities to improvement in the approval dimension. Requirements Management aims to maintain the quality level in all three dimensions.

EU 9.2 Parameters of the Requirements Engineering Process (K2)

The RE process can be very varied and must adapt itself in particular to the given constraints. In all the various existing RE processes in different approaches, there is only a certain number of process parameters that can be changed when selecting or adjusting the RE process:

- Timing of the elicitation,
- Level of detail of the documentation, i.e. heavyweight versus lightweight specification,
- Incorporation of changes, in particular: Change Request versus Product Backlog,
- Allocation of responsibility.

These parameters should be selected according to the constraints. Such constraints are:

- The size of the project.
- Is it a new implementation or a small enhancement, improvement or variation to an existing, mature system or product?
- Was a fixed price agreed or not?
- Is there a stable team that has been working together for years?
- Availability of people and their qualifications.

Timing of the Elicitation (upfront or iterative)

Requirements can either be determined completely at the beginning of the project (upfront) or iteratively (iterative Requirements Engineering): In the first case (upfront), a requirements specification (e.g. a specification sheet) is created at the beginning of the project, which completely describes the planned project scope. With iterative Requirements Engineering, one does not aim to define the requirements, or even just the project scope, completely at the beginning, but rather considers the requirements documentation (e.g. the Product Backlog) as a preliminary list. Requirements can be added or changed at any time, even during implementation.

Note: There is a difference between iterative Requirements Engineering and iterative development. It is therefore conceivable to first create a complete requirements specification upfront and later implement the requirements through iterative development.

If the project is a small enhancement, improvement or variation of an existing, mature system or product, then it is to be expected that stable requirements can be defined for the entire project with few surprises expected. This is where upfront requirements determination is possible and useful.

However, if the project is very innovative with many uncertainties, in a volatile environment, has undecided or conflicting stakeholders or there are other risk factors that make a reliable upfront specification impossible, iterative Requirements Engineering serves to reduce risk.

Level of Detail of Requirements Documentation

The level of detail of the documentation or specification can vary between heavy and lightweight requirements: heavy specification describes all requirements in detail, including all their attributes and traceability relationships, making the specification very comprehensive.

Lightweight specification describes requirements only as comprehensively as necessary and not earlier than necessary. When certain information is required depends on the process model. What is needed depends on the stakeholders, their needs and background. A project-specific stakeholder analysis helps to define how detailed the requirements specification must be.

Among other things, the purpose of a specification is to enable the developer to understand what stakeholders want. Details of the implementation are either left to the developer (especially if he is very familiar with the domain), discussed verbally without documenting or refined using a prototype. Lightweight requirements specification describes requirements as user stories, for example. Requirements are only specified in detail when their implementation is about to begin.

Even though upfront specification is usually heavyweight (e.g. in the waterfall model and V-Modell XT) and iterative lightweight (as in Scrum and other agile methods, see EU 10), the two parameters timing and level of detail are independent of one another. It is possible to create both a lightweight specification upfront and a heavyweight one iteratively (as in the Rational Unified Process).

Change Management: Incorporation of Changes (Change Request versus Product Backlog)

Requirements change during the project. Some RE processes integrate new or changed requirements as change requests in the requirement specification and development process. These are usually projects with a fixed price and upfront requirement specification, i.e. requirements determination is completed at a certain point in time from an organizational and legal point of view. From a legal point of view, later changes are contractual changes. A change request legally means a new contract. Normally, the contract already specifies how changes are to be handled. They usually go through a simplified approval procedure with the steps: analysis (of requirements and their benefits), impact analysis (i.e. analysis of changes to the system, their costs and risks), decision by the Change Control Board and then implementation. A change request is often described using a change request template that assigns it a unique number and title, describes the problem to be solved and the proposed solution, quantifies costs, benefits and risks, and manages the status (requested, accepted, rejected, postponed, implemented).

In iterative Requirements Engineering, however, requirements are collected in the Product Backlog and all requirements - old and new - are treated equally. This is made possible by the fact that one never commits oneself to a defined system scope. Nevertheless, it is not mandatory for an upfront requirements specification to treat later requirements as change requests.

It would be conceivable to adjust the requirements artifact created upfront later without recording and approving changes as change requests. Changes to the requirements artifacts must of course be documented and traceable.

Allocation of Responsibility

A single role (for example, the Requirements Manager) can be responsible for the RE process in the sense that he or she plans, controls and improves the RE process.

He or she may also carry out all activities of the RE process him/herself. However, there can also be an entire team or several roles responsible for Requirements Engineering, either for different activities or different content (e.g. functional requirements versus usability or usefulness requirements). Requirements Engineering can also be closely integrated into the development process without a separate RE process or the role of Requirements Analyst. In this case, the development team carries out the Requirements Engineering activities, i.e. the team members collect, document, check and manage requirements.

EU 9.3 Documenting the Requirements Engineering Process (K2)

The RE process consists of numerous activities of the four types mentioned above, such as elicitation workshops, specification reviews, etc., as explained in the Foundation Level syllabus [IREB FL 2012]. Many of these activities are planned in the form of meetings, as frequently many people are involved. The order of these activities results from the choice of process parameters (see EU 9.2), which may be defined project-specifically or also company-wide. The activities and their sequence can be presented as a UML activity diagram. The activity diagram can also show the assignment of activities to roles.

The assignment of responsibilities for activities to roles can also be presented in more detail using an RACI matrix like the following. RACI stands for:

- R = responsible = responsible for the execution
- A = accountable = accountable, i.e. one authorizes for example the activity and its budget
- C = consulted = (will be) consulted, especially in terms of technical, content-related responsibility
- I = informed = to be informed, i.e. the person is to be informed

Table 3 shows an example of an excerpt from an RACI matrix.

Activity	**Requirements Engineer**	**Project head**	**Key User**
Document Analysis: Handbook of the legacy systems	R, A	I	
Creativity workshop with key users	R	A	C
…			

Table 3: Example of a RACI matrix for Requirements Engineering.

To manage dates and budgets quantitatively, the RE process can also be presented as a project plan.

Other documents that can represent and support the RE process are: Project plan, checklists, templates, sample documents and guidelines.

If many people are involved in the RE process, it also makes sense to support this process with a tool. All Workflow Management Systems in the broadest sense are suitable for this.

EU 9.4 Monitoring and Controlling the Requirements Engineering Process (K1)

Monitoring the RE process means ensuring that all activities are carried out and the defined results are delivered on time and that the activities remain within budget. Reports that regularly record dates, consumed budget, status and percentage of completion of the RE process and its individual activities and compare the actual values with the target values from planning are helpful for this (see EU 8).

Controlling the RE process means executing it according to the plan or, if the process deviates from the plan, taking corrective action. For example, if it becomes apparent that the deadline or budget cannot be met, the consequences for the overall project must be determined and - if appropriate - countermeasures taken. To adjust the ongoing process to the plan, planned activities may have to be omitted, brought forward or performed with less effort. Careful trade-offs must be made where they cause the least damage, e.g. individual stakeholder groups are not interviewed, individual open questions are not clarified, details are not specified, unimportant change requests are rejected and so on. It is important to consider the risk: Does the benefit of the savings outweigh the possible damage?

EU 9.5 Process Improvement in the Requirements Engineering Process (K2)

A process can still be further improved. The basis for every process improvement is the analysis and documentation of the current status (see EU 9.3). The current status of Requirements Management can be documented in a Requirements Management Plan (RMP). It describes the Requirements Management of a company or project: the RE process, the Requirements Information Model, the attributes and views, and all other specifications described in this syllabus. The RMP serves as documentation of the actual state as a basis for reflecting the current approach and can also structure the planned process improvement.

A process improvement can be carried out either abruptly - a process rearrangement - or continuously. A process rearrangement changes many activities and parameters of the process at the same time. This has the advantage that it is possible to achieve a significant increase in efficiency, which, however, usually only occurs after all participants have become accustomed to the new process. However, there is also the risk that the new process will not prove its worth and will reduce efficiency. Resetting will then again involve great effort.

Continuous process improvement avoids this risk and leads to short-term improvements with little effort. According to the principle of Continuous Process Improvement (CPI), processes are gradually optimized by repeating the following four activities (PDCA) iteratively:

- **Plan**: The actual process and, in particular, the need for improvement are analysed. Based on this, the desired process is planned and documented.
- **Do**: Improvement actions are developed and tested in a pilot project and accompanied by measurements.
- **Check**: One checks whether the actions have brought about the desired improvement. The actual values are compared with the planned values.
- **Act**: Based on the results of the actual plan analysis, improvement actions are introduced continuously or, if necessary, new actions are planned. The implementation of actions is monitored and accompanied by measurements.

The actual and target process is characterized using measured quantities (see EU 8). Such measured quantities can be:

- The proportion of the project budget invested in Requirements Engineering. Both too much and too little can be questionable. Normally it is 10-30% of the project budget.
- The number of requirements still to be implemented (weighted according to expected effort). It measures the work still to be done before the end of the project.
- Burndown rate or velocity, i.e. the number of requirements that are implemented per time unit weighted according to effort. Together with knowledge of the requirements still to be implemented, forecasts can thus be made about the remaining duration of the project.
- Change Rate of Requirements. A rate of 1-5% of the requirements per month (measured in effort) and 30-50% over the project duration is considered normal [Ebert 2012]. Fewer changes may mean that no one is really interested in the requirements and stakeholders are not sufficiently involved. Too many changes are also an alarm signal: Requirements are not yet stable, stakeholder groups may be too heterogeneous or in conflict, and it is still too early to implement the requirements.
- Throughput time of change requests from order to implementation.

With the help of benchmarking it is possible to find out which figures are meaningful and achievable as target values.

Improvement actions can either refer to the process parameters described in EU 9.2 or to how the individual activities are carried out in detail, e.g. with the help of which methods.

Another possibility for process improvement is to analyze the errors made in Requirements Engineering, e.g. defects found during the specification inspection, or defects delivered with the system that can be traced back to Requirements Engineering. Then one asks about their causes and the causes of the causes. This gives ideas for improvement actions.

Maturity models such as CMMI [CMMI 2011] or ITIL [Beims 2012], [Ebel 2014] offer more concrete help for process improvement in Requirements Engineering (but not only there). They describe activities or practices that must be carried out to reach a certain level of maturity. Introducing new activities and practices that have not yet been implemented represents then a process improvement. All other methods of process improvement can also be used, such as TQM (Total Quality Management) and Six Sigma.

In particular, the improvement of Requirements Engineering supports the collection of best practices by Sommerville and Sawyer [Sommerville & Sawyer 1997].

Action Plan [Wiegers 2005, p. 66] supports the concrete planning of process improvement. Such an action plan contains the following:

- Name of the improvement project,
- Date,
- Goals (of improvement, expressed as business goals),
- Indicators of success (i.e. achievement of goals),
- Organizational influence of change,
- Participants (employees, their roles and time budgets),
- Measurement and reporting process (when will the progress of actions within this plan be monitored, by whom, and how),
- Dependencies, risks and boundary conditions,
- Estimated completion date of all actions within this plan,
- Actions (3-10 per plan) with responsible person, target date, purpose, description, delivery items and resource requirements.

When improving the RE process, it should be noted that it cannot be optimized on its own, but only in cooperation with other project activities such as Project Management, development and testing. Changes in the RE process will also affect those people's work.

EU 10 Requirements Management in Agile Projects (K1)

Duration: 1 ½ hours
Terms: User Story, Sprint, Product Backlog, Burndown-Chart

Educational Objectives:
EO 10.1 Knowing the basic principles of Agile Software Development (K1)
EO 10.2 Knowing key activities and artifacts in Requirements Management in Agile projects (K1)
EO 10.3 Knowing how RM activity is mapped to Scrum activities (K1)

EU 10.1 Background (K1)

Classical approaches to software development, and thus also to Requirements Engineering, emphasize a plan that reaches far into the future. At the end of the 1990s and the beginning of the 2000s a counter-movement to this emerged. This moved away from long-term plans and placed emphasis on short-term plans with many feedback loops. The classification by [Boehm & Turner 2003] is based on this difference: the two groups are called "Agile" (as proposed in the Agile Manifesto) and "plan-driven" (in the original: plan-driven). In general, the Agile Manifesto is the common foundation of all Agile approaches. From the software developers' point of view, the Agile Manifesto states [AgileManifesto]:

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

These values show that the Agile approach is more about team collaboration, productivity and individual strengths than about contracts and documentation. This distinguishes Agile methods from plan-driven approaches that require clear contractual elements (e.g. project scope, release plans or a defined change process).

EU 10.2 Requirements Management in Agile Projects (K1)

Both Agile and plan-driven approaches are inherently heterogeneous (see e.g. [Linssen 2009], [Korn 2013]). Therefore one cannot state in general how requirements are handled in agile projects. The following statements are therefore not to be understood as complete or comprehensive. The following section primarily uses Scrum [Sutherland & Schwaber 2013] as a typical representative of Agile approaches.

Scrum is a "framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value" [Sutherland & Schwaber 2013].
The most discussed form of requirement in Agile projects is the user story. A user story shows the requesting role, the requirement itself and the benefit. A user story will be the responsibility of the Product Owner (PO). Out of this, the development team (abbreviated as DT) derives concrete work instructions (tasks) for a development cycle (sprint).

A user story usually has the following form:

> As <ROLE> I want <TARGET/WISH>, so that <BENEFIT>.

Specifying the benefit is optional. Acceptance criteria and test cases are specified more precisely for a user story. They have the following form:

> Under the condition that <PRECONDITION>, if <TRIGGER> then < RESULT>.

User stories are often annotated with the implementation estimate ("How complex is the implementation of the user story?") and its value contribution to the project/vision. User stories are usually sorted by their planned implementation order. User stories can be grouped, e.g. in epics (a user story that is too big to be implemented alone and therefore shared) or in themes (user stories with similar content).

Most Agile approaches do not strictly specify which artifacts to work with. The Use Case, which is also well-established as a form of expression in classical software development, can be used as an artifact in an Agile environment as well (e.g. [Cockburn 2001] or [Jacobson el al. 2011]).

Artifacts (e.g. user stories or use cases) are often evaluated and prioritized in a Product Backlog. In Scrum this is the responsibility of the Product Owner. The Product Owner is the person who manages the artifacts to be implemented. To do so, the Product Owner prioritizes artifacts in the Product Backlog, with a priority based on the value contribution of the artifact and the needs of the stakeholders. The implementation order of the artifacts results on the one hand from their priority and on the other hand from technological dependencies.

Written reporting is minimized in Agile approaches through highly interactive work: the communication of all parties involved should reduce the effort required for written reports. Regular meetings ("events") are used for this purpose.

Again, this varies between the different approaches. In Scrum, daily meetings of the development team ("Daily Standup"), reviews and retrospectives will be used. During Reviews the result of a development cycle (in Scrum: "Sprint") will be presented and approved by the Product Owner. In a Retrospective, the interaction of a team will be evaluated retrospectively. Written reports include Burndown Charts ("How much work still needs to be done in this sprint?") or an Impediment Backlog ("What hinders the team in its work?"). The Scrum Master (abbreviated SM) ensures that these meetings are held in accordance with rules.

While the term process is rarely used in Agile approaches, the guidelines of the approaches imply a concrete workflow. Improvement of these processes will be achieved through regular retrospectives.

Scaling Agile approaches to large and distributed teams is in its infancy, currently some frameworks are being developed. Some approaches can be found in [Eckstein 2004], [Eckstein 2010], [Leffingwell 2011] and [Korn & Berchez 2013].

EU 10.3 Mapping RM Activities to Scrum Activities (K1)

Scrum only specifies general work processes. In Table 4 the RM activities are assigned to the Scrum activities or artifacts. Furthermore, the executing role is specified in Scrum. Not all RM activities are covered in the Scrum Guide. Beside the Scrum Guide there is a not insignificant amount of literature describing more or less successful additions to Scrum. Whether and how the corresponding RM activity is then executed in a Scrum project is up to the Scrum team.

Requirements Management activity	**Scrum activity or artifact**	**Scrum role**
Attribute	User Stories in the Backlog: Description, order, estimate, status and value. Optional: Grouping	PO, DT
Evaluation and prioritisation	1. Estimation of the benefits and costs through Planning Poker 2. Arrangement of User Stories in the Product Backlog 3. Selection of User Stories for a Sprint 4. Prioritization within a Sprint	1. DT 2. PO 3. PO and DT 4. DT
Traceability	There is an implicit traceability of User Stories to the corresponding acceptance test cases and, with suitable attribute assignment, back to the sources of the User Stories. In addition, traceability is possible within the Product Backlog (dependencies) and from User Stories to the source code. Scrum says nothing about connecting User Stories within the Product Backlog. Traceability via epics (grouped User Stories) would be conceivable. Traceability will be documented only if necessary.	None
Versioning of requirements	Versioning of User Stories is unnecessary. The current version of a User Story is always relevant.	None or PO

Requirements Management activity	Scrum activity or artifact	Scrum role
Changes	Changes can be proposed at any time. New requirements lead to new User Stories, requirement changes to the fact that a User Story is changed or replaced by a new one.	PO
Variant Management	Agile methods do not explicitly support variant management. However, it is possible to use standard methods of variant management.	PO
Reporting	Reports are mainly verbal. The artifacts used to track the completion status can also serve as reports: • Daily Standup • Sprint Review • Sprint Retrospective • Product backlog • Sprint backlog • Burndown chart	DT
Process Management	Sprint Retrospective and Impediment Backlog	SM, DT

Table 4: Mapping of Requirement Management activities to Scrum activities.

EU 11 Use of Tools in Requirements Management (K1)

Duration: 0,75 hours
Terms: Tool, tool selection, data exchange

Educational Objectives:
EO 11.1 Knowing the role of tools in Requirements Management and their core functions (K1)
EO 11.2 Knowing the basic procedure for tool selection for Requirements Management tools (K1)
EO 11.3 Knowing the need for cross-tool data exchange (K1)

EU 11.1 Role of Tools in Requirements Management (K1)

The use of tools is intended to make it easier for the Requirements Manager to document and manage requirements. Due to their special functionalities, Requirements Management tools enable a holistic view of requirements, in that, among other things, relationships between different requirements as well as the life cycle of individual requirements can be represented.

A Requirements Management tool is a software application whose main objective is to support activities in Requirements Management.

Many applications are traditionally used in software and system development. Many of them cover some aspects of Requirements Engineering and/or Requirements Management. The distinction between these tools and those for Requirements Management is therefore not always clear-cut.

Tools for Requirements Management are often based on assumptions about the development method. Some tools are designed for specific...

- Process models (e.g. Agile development, prototyping or product management)
- Application domains (e.g. automotive industry, medical engineering or defence systems)
- Work environments (e.g. virtual office, local collaboration or global collaboration).

This results in a variety of Requirements Management tools that often support only some of the tasks of a Requirements Manager.

Central functions of a Requirements Management tool are [Sommerville & Sawyer 1997]:

- Editor for requirements including their attributes
- Import of requirements from existing documents into the tool and export of managed requirements to other formats
- Tracking of requirements
- Versioning of requirements and creating configurations
- Creating views of requirements

As described in [Rupp & Sophist 2009], it is important to ensure that the Requirements Management tool fits the procedures and processes established in the company. Once the process and the basic data model for the company have been created, a tool evaluation can be carried out.

EU 11.2 Basic Procedure for Tool Selection (K1)

Based on [Rupp & Sophist 2009], the following steps are necessary when selecting a tool for Requirements Management:

- Launching a tool selection project
- Defining rough selection criteria by formulating basic requirements.
- Carrying out the rough selection (long list) to identify the first potential systems.
- Refining the catalogue of criteria on the basis of new and refined requirements for the tool.
- Conducting a fine selection (shortlist), up to a favored software candidate.
- If no tool precisely meets requirements, it is necessary to adapt (customize) the software application.
- In order to strengthen the acceptance in the company and to eliminate possible last concerns, a pilot project will finally be launched.

In case the pilot project reveals that the selected tool does not provide the desired support, the tool selection must be repeated. If no tool exactly meets the requirements, a process adjustment can be made instead of adjusting a tool.

EU 11.3 Data Exchange between Requirements Management Tools (K1)

When several companies work together on a project, it is often necessary to exchange requirements between different Requirements Management tools.

This is the case, for example, if a supplier wants to further refine the customer's requirements within the scope of the system development or to establish traceability to subsequent development artifacts such as design or test cases.

Exchanging information via standard document formats, as can be typically exported and imported, is insufficient as relevant information such as a unique identifier, the editor, structural information or versioning information about the requirements is usually lost. Above all, such an approach does not sufficiently support an exchange of updates (e.g. after changes have been incorporated).

To enable a tool-independent exchange of requirements while retaining central requirement characteristics, the industry standard Requirements Interchange Format (ReqIF) was created and standardized by the Object Management Group.

Essential advantages, which result from the use of ReqIF, are thereby:

- With ReqIF, collaboration between companies can be improved by applying Requirements Management methods across companies.
- Partners do not have to work with the same tool. Suppliers do not need to have their own Requirements Management tool for each customer.
- Requirements can be transferred within an organization, even across tool boundaries.
- With ReqIF, requirements with all attributes and meta information can be exchanged without loss, unlike document exports to Word, PDF, etc.

Currently, a variety of tools support the ReqIF format.

References

[AgileManifesto] Manifesto for Agile Software Development. Available at http://agilemanifesto.org/.

[Atkinson 2002] C. Atkinson: Component-based Product Line Engineering with UML. Addison-Wesley, 2002.

[Basili et al. 1984] V. Basili, G. Caldiera, and H. D. Rombach: The Goal Question Metric Approach. In: Encyclopedia of Software Engineering. John Wiley & Sons, 1994, S. 528–532.

[Beims 2012] Martin Beims: IT-Service Management mit ITIL®: ITIL® Edition 2011, ISO 20000:2011 und PRINCE2® in der Praxis, Carl Hanser Verlag GmbH & Co. KG, 3., aktualisierte Auflage, 2012.

[Boehm & Turner 2003] B. Boehm und R. Turner: Using Risk to Balance Agile and Plan-Driven Methods. In: IEEE Computer, Volume 36, Issue 6, June 2003, pp. 57-66.

[Boutkova 2011] E. Boutkova: Experience with Variability Management in Requirement Specifications. In: D.E. Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid (eds): Software Product Lines – 15th International Conference (SPLC), München, 2013, pp. 303-312.

[Boutkova & Houdek 2011] E. Boutkova and F. Houdek: Semi-automatic identification of features in requirement specifications. In: Proceedings of the 19th International Requirements Engineering Conference, Trento, Italy, September 2011, pp. 313-318.

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

[Czarnecki & Eisenecker 2000] K. Czarnecki and U.W. Eisenecker: Generative Programming: Methods, Tools, and Applications. Addison-Wesley, 2000.

[Cleland-Huang et al. 2013] J. Cleland-Huang, R. S. Hanmer, S. Supakkul, and M. Mirakhorli: The Twin Peaks of Requirements and Architecture. IEEE Software, vol. 30, no. 2, pp. 24-29, March-April, 2013

[Clements & Northrop 2007] P. Clements and L. Northrop: Software Product Lines: Practices and Patterns. Addison-Wesley, Boston, 6. Auflage, 2007.

[CMMI 2011] CMMI® for Development, version 1.3. CMU/SEI-2010-TR-033. Available at http://cmmiinstitute.com/wp-content/uploads/2013/01/10tr033de_v11.pdf

[Coplien et al. 1998] J. Coplien, D. Hoffmann, and D. Weiss: Commonality and Variability in Software Engineering. In: IEEE Software, Volume 15, Issue 6, 1998, pp. 37-45.

[Davis 2003] A. Davis: The Art of Requirements Triage. In: IEEE Computer, Volume 36, Issue 3, 2003, pp. 42-49.

[Davis 2005] A. Davis: Just Enough Requirements Management, Dorset House, 2005.

[Ebel 2014] N. Ebel: ITIL®(R) 2011 Edition: Grundlagen und Know-how für das IT Service Management und die ITIL®(R)-Foundation-Prüfung, dpunkt.verlag GmbH, 1. Auflage, 2014

[Ebert 2012] C. Ebert: Systematisches Requirements Engineering. Dpunkt, 4. Auflage, 2012.

[Eckstein 2010] J. Eckstein: Agile Software Development with Distributed Teams. Dorset House Publishing, 2010.

[Eckstein 2004] J. Eckstein: Agile Software Development in the Large. Dorset House Publishing, 2004.

[Gabler 2014] Springer Gabler Verlag (publisher), Gabler Wirtschaftslexikon, keyword: product line, online on the Internet: http://wirtschaftslexikon.gabler.de/Archiv/56903/produktlinie-v6.html (as of: 11. Nov 2014)

[Glinz 2008] M. Glinz: A Risk-Based, Value-Oriented Approach to Quality Requirements. IEEE Software, Nr. 2, S. 34-41, 2008.

[Gotel & Finkelstein 1994] O.C.Z. Gotel and A.C.W Finkelstein: An Analysis of the Requirements Traceability Problem. In Proceedings of IEEE International Conference on Requirements Engineering, 1994, pp. 94-101.

[Hull et al. 2011] E. Hull, K. Jackson, and J. Dick: Requirements Engineering. Springer, 3. Auflage, 2011.

IEC DIN 61508] IEC DIN EN 61508-2 Functional safety of safety-related electrical/electronic/programmable electronic systems. VDE Verlag, 2002.

[IREB FL 2012] Syllabus IREB Certified Professional for Requirements Engineering - Foundation Level, Version 2.1, 2012.

[IREB Glossary] M. Glinz: A Glossary of Requirements Engineering Terminology. Version 1.6, May 2014.

[ISO 12207] International Organization for Standardization (ISO): ISO/IEC 12207: 2008, Systems and software engineering — Software life cycle processes. 2008.

[ISO 26262] International Organization for Standardization (ISO): ISO 26262, Road vehicles -- Functional safety. 2011.

[ISO 26550] International Organization for Standardization (ISO): ISO/IEC 26550:2013: Software and systems engineering -- Reference model for product line engineering and management. 2013.

[ISO 9000] International Organization for Standardization (ISO): ISO 9000:2005: Quality management systems — Fundamentals and vocabulary. 2005.

[Jacobson et al. 2011] I. Jacobson, I. Spence, and K. Bittner: Use Cases 2.0. Ivar Jacobson International, 2011.

[Kang et al. 1990] C. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson: Feature-Oriented Domain Analysis (FODA) - Feasibility Study. Software Engineering Institute, 1990.

[Karlsson & Ryan 1997] J. Karlsson and K. Ryan: A Cost-Value Approach for Prioritizing Requirements. IEEE Software 14, Nr. 5, S. 67–74, 1997.

[Korn & Berchez 2013] H.-P. Korn and J.P. Berchez (eds.): Agiles IT-Management in großen Unternehmen. Symposion, 2013.

[Küpper 2005] H.-U. Küpper: Controlling: Konzeption, Aufgaben, Instrumente. Schäffer-Poeschel, 4. Auflage, 2005.

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

[Loucopoulos & Karakostas 1995] P. Loucopoulos and V. Karakostas: System Requirements Engineering. McGraw-Hill, 1995.

[Maeder et al. 2009] P. Mäder, O. Gotel, and I. Philippow: Getting Back to Basics: Promoting the Use of a Traceability Information Model in Practice. In: Proceedings of 5th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE2009), Vancouver, Canada, May 2009, pp. 21-25.

[Maeder et al. 2013] P. Mäder, P.L. Jones, Y. Zhang, and J. Cleland-Huang: Strategic Traceability for Safety-Critical Projects. In: IEEE Software, Volume 30, Issue 3, May / June 2013, pp. 58-66.

[Moisiadis 2002] F. Moisiadis: The fundamentals of prioritizing requirements. In: Systems Engineering, Test & Evaluation Conference, Sydney, October 2002.

[Nemnich 2006] B. Nemnich: Der Sarbanes-Oxley Act und die Wirkung auf die IT. Diplomarbeit, Grin Verlag, 2006.

[Nuseibeh 2001] B. Nuseibeh: Weaving the Software Development Process between Requirements and Architecture. In: Proc. of ICSE2001 Workshop STRAW-01, Toronto, May 2001.

[OMG 2013] Requirements Interchange Format (ReqIF). Object Management Group, Version 1.1., 2013, available at http://www.omg.org/spec/ReqIF/1.1/PDF/

[PMI 2013] PMI: Project Management Book of Knowledge (PMBOK). Project Management Institute, 5th Ed., 2013.

[Pohl, 1994] K. Pohl: The Three Dimension of Requirements Engineering: A Framework and its Application. Information Systems 3, 19 (June 1994), pp. 243–258.

[Pohl 1996] K. Pohl: Process-Centered Requirements Engineering. John Wiley Research Science Press, 1996.

[Pohl 2010] K. Pohl: Requirements Engineering – Fundamentals, Principles, Techniques. Springer, 2010.

[Pohl et al. 2005] K. Pohl, G. Böckle, and F. van der Linden: Software Product Line Engineering - Foundations, Principles, and Techniques. Springer, 2005.

[Pohl & Rupp 2011] K. Pohl and Chris Rupp: Requirements Engineering Fundamentals. Rocky Nook, 2011.

[Robertson & Robertson 2014] S. Robertson, J. Robertson: Mastering the Requirements Process – Getting Requirements Right. Addison-Wesley, 3rd Edition, 2014.

[Rupp & Sophist 2009] C. Rupp & die SOPHISTen: Requirements-Engineering und –Management, Hanser, 5. aktualisierte und erweiterte Auflage, 2009.

[SEI 1999] Carnegie Mellon SEI (1999): The Capability Maturity Model, Guidelines for Improving the Software Process. Addison-Wesley, 1999.

[SEI 2010] Carnegie Mellon SEI (2010): CMMI for Services, Version 1.3, Improving processes for providing better services, 2010.

[Schienmann 2001] B. Schienmann: Kontinuierliches Anforderungsmanagement Prozesse - Techniken – Werkzeuge. Addison-Wesley, 2001.

[Schobbens et al. 2006] P.-Y. Schobbens, P. Heymans, and J.C. Trigaux: Feature Diagrams: A Survey and a Formal Semantics. In: Proceedings of the 14th International Requirements Engineering Conference (RE'06), September 2006, pp. 139-148.

[Sommerville & Sawyer 1997] I. Sommerville and P. Sawyer: Requirements Engineering: A Good Practice Guide. John Wiley & Sons, 1997.

[Sutherland & Schwaber 2013] J. Sutherland and K. Schwaber: Scrum Guide, July 2013, available at www.scrum.org

[US Congress 2002] US Congress: Sarbanes-Oxley Act. Washington, USA, 107th Congress of the United States of America, 23.01.2002.

[Van Lamsweerde 2009] A. van Lamsweerde: Requirements Engineering – from System Goals to UML Models to Software Specifications. John Wiley and Sons, 2009.

[Weiss & Lai 1999] D. Weiss and C. Lai: Software product-line engineering: a family-based software development process. Addison-Wesley, 1999.

[Wiegers 2005] Karl Wiegers: Software Requirements, Microsoft Press Deutschland, 1. Auflage, 2005.

[Wiegers & Beatty 2013] K. Wiegers and J. Beatty: Software Requirements, 3rd Edition. Microsoft Press, 2013.

[Young 2004] R. Young: The Requirements Engineering Handbook, Artech House, Boston, 2004.

[Ziegbein 1998] K. Ziegbein: Controlling. Kiehl Friedrich Verlag, 6. Auflage, 1998.

Version History

Date	*Version*	*Author*	*Description of the change*
September 11, 2019	1.1.0		Initial version based on German version 1.1.0