Syllabus

REQB®

Certified Professional for Requirements Engineering

Foundation Level

Version 2.1

2014

Overview of Changes

Version	Date	Comment
0.1	Apr. 17, 2006	First version of the syllabus; creation of a basic structure for the syllabus
0.2	Jul. 20, 2006	Expansion to Version 0.1
0.3	Sep. 4, 2006	Further expansion and revision of Version 0.1
0.4	Oct. 10, 2006	Revised Version 0.3
0.5	Dec. 15, 2006	Revised Version 0.4
0.6	Feb. 7, 2007	Completely revised Version 0.5
0.7	Apr. 10, 2007	Revised version for review
0.8	Jun. 15, 2007	Alpha version
0.9	Sep. 1, 2007	Beta version
1.0	Jan. 15, 2008	Released Version 1.0
1.1	May 29, 2008	Updated Version 1.1
1.2	Jul. 1, 2008	Updated Version 1.2
1.3	May 15, 2011	Updated Version 1.3
1.4	Jun. 30, 2013	Changes resulting from review of Version 1.4
1.5	Aug. 19, 2013	Changes: changed and reduced LOs, added mapping between BOs and LOs, changes in the structure of chapter 3 – RE
1.6	Oct. 20, 2013	Reduced number of BOs, changed requirements classification, changed definition of conceptual model, more requirements on exercises added, reduced list of literature, incorporated comments from the review of Version 1.5
1.7	Nov. 01, 2013	Minor changes after the last cycle of review.
2.0	Dec. 01, 2013	Final version.

2.1	Feb 09, 2014,	Minor changes

Main Idea

The central theme for this syllabus was that the complexity of systems and software and our dependency on them continue to increase. The result is a high level of dependency on the freedom from errors in the software, hardware or other products. The Requirements Engineering Qualifications Board (REQB) has therefore decided to create uniform international standards in the area of Requirements Engineering. Standards are like languages – it is only if you understand them that you can work effectively. In order to create a uniform language in this important area of requirements engineering, international experts got together in REQB and developed this syllabus.

Acknowledgements

This document was produced by a core team from the Requirements Engineering Qualifications Board Working Party Foundation Level (Edition 2013): Karolina Zmitrowicz (chair), Sergey Kalinov, Beata Karpinska, Andrey Konushin, Salvatore Reale, Alexander Lindner, Ingvar Nordström, Alain Ribault.

The core team thanks all the National Boards for their suggestions and input.

Table of Contents

Introduction

Purpose of the Syllabus

This syllabus defines the Foundation Level of the training program to become a REQB Certified Professional for Requirements Engineering (CPRE). REQB developed this syllabus in cooperation with the Global Association for Software Quality (GASQ). The scope of the REQB program covers the process of Requirements Engineering for all types of IT-related products consisting of software, hardware, services and business processes, and documentation as well.

REQB provides this syllabus as follows:

1. To National Boards, to translate into their local language and to accredit training providers. Translation may include adapting the syllabus to the particular language needs and modifying the references (books and publications) to adapt to the local publications.
2. To Exam Boards, to allow creating examination questions in their local language adapted to the Learning Objectives defined in this syllabus.
3. To training providers, seeking accreditation as REQB recognized training providers, to produce courseware. All areas of this syllabus must correspondingly be incorporated in the training documents.
4. To certification candidates, as preparation material for the certification exam (as part of an accredited training course or independently).
5. To the international Requirements Engineering community, to advance the profession of a Requirements Engineer.

Examination

The examination to become a Certified Professional for Requirements Engineering Foundation Level is based on this syllabus. All sections of this syllabus can thereby be addressed by exam questions. The examination questions are not necessarily derived from an individual section; a question may refer to several sections.

The format of the examination questions is Multiple Choice.

Examinations can be taken after having attended accredited courses or in open examination sessions (without a previous course). You will find detailed information regarding examination

times on REQB's website (www.reqb.org), on the website of GASQ (www.gasq.org) or on the website of your local examination provider in your country.

Accreditation

Providers of a REQB Certified Professional for Requirements Engineering Foundation Level course must be accredited by the Global Association for Software Quality. Their experts review the training provider's documentation for accuracy and compliance with the content and Learning Objectives of the syllabus. An accredited course is regarded as conforming to the syllabus. At the end of such a course, an officially Certified Professional for Requirements Engineering examination (CPRE exam) may be carried out by an independent certification institute (according to ISO 17024 rules).

Accredited Training Providers can be identified by the official REQB Accredited Training Provider logo.

Internationality

This syllabus was developed in cooperation between international experts. The content of this syllabus can therefore be seen as an international standard. The syllabus makes it possible to train and examine internationally at the same level.

Business Benefits

Objectives, benefits and the main focus of the REQB Certified Professional for Requirements Engineering Foundation Level program are presented in the following table (Table 1 Objectives of the Foundation Level program, its benefits and main focus).

Objectives	
Obtain new key qualification	Any software, hardware or service solution is based on stakeholders' requirements and aims to satisfy specific business needs. To be able to deliver a solution compliant with the needs of the target group, proper Requirements Engineering is necessary. Managing the requirements and developing them into the solution design is very important, but to achieve full success and deliver the best solution, the quality should be considered as well. All those aspects are covered in the Foundation Level.
Increase your customers' satisfaction	Customer satisfaction is achieved when they experience the solution meeting their expectations and needs. Improved Requirements Engineering minimizes discrepancies between the expectations and the perception of conformance. Higher quality of the final product also strengthens customer loyalty.
Minimize development and follow-up costs	Proper Requirements Engineering minimizes the project and product risks and helps avoid costs related to rework resulting, for example, from discrepancies between the customer's expectations and the efficiency of the solution being developed. Well managed requirements also reduce costs for future improvements or corrective actions.
Competitive advantage	Requirements Engineering helps to deliver better products or services, meeting all business needs and expectations. Better products or services help to obtain the confidence and loyalty of desirable target groups and increase competitive advantage.

Objectives	
Focus	
Requirements	Understanding the basic concepts related to requirements, their classification and levels of abstraction; explaining the meaning of requirements attributes and the role of quality criteria.
Requirements Engineering process	Explaining the Requirements Engineering process, its activities, actors, and deliverables, most important principles and best practices for developing and managing requirements of software/hardware/service products.
Standards, norms and legal regulations	Overview of the most important standards, norms and legal regulations and their effect on Requirements Engineering.
Requirements Engineering in development and maintenance models	Explaining the principles of adjusting the generic Requirements Engineering process to specific process models.
Tools and techniques	Explaining the usage and benefits of techniques to identify requirements; problem and solution modeling; quality assurance and control techniques; tool support for the Requirements Engineering process.
Exercises	Writing good requirements; quality control of requirements; identification and analysis of requirements.

Table 1 Objectives of the Foundation Level program, its benefits and main focus

The Foundation Level of the REQB Certified Professional for Requirements Engineering program is suitable for all persons involved in product/business solution development and maintenance, including business and system analysts, marketing teams, hardware/software designers, GUI designers, project managers, customer representatives, maintenance and technical staff, IT auditors and quality assurance representatives.

The main purpose of the Foundation Level program is to provide a common terminology and a common understanding of the key concepts related to the Requirements Engineering process. A knowledge base provided in the Foundation Level program supports common definitions and concepts related to Requirements Engineering and it explains the Requirements Engineering process together with its deliverables. The program is based on generally accepted standards and best practices.

Learning Objectives

The Learning Objectives of this syllabus have been divided into different cognitive levels of knowledge (K-levels). This makes it possible for the candidate to recognize the "knowledge level" of each point.

Each section of this syllabus has a cognitive level associated with it:

- K1 - Proficiency/Knowledge: Knowledge of precise details such as terms, definitions, facts, data, rules, principles, theories, characteristics, criteria, procedures. Students are able to recall and express knowledge.
- K2 - Understanding: Students are able to explain or summarize facts in their own words, provide examples, understand contexts, interpret tasks.
- K3 - Apply: Students are able to apply their knowledge in new specific situations, for example, by applying certain rules, methods or procedures.

Business Outcomes

After completing the REQB Certified Professional for Requirements Engineering Foundation Level program, a person can:

- BO01 Communicate the fundamental concepts of Requirements Engineering, requirements and their role for a product life cycle.
- BO02 Create awareness of the meaning of the Requirements Engineering process and its deliverables.
- BO03 Communicate the main activities and purpose of the process of Requirements Engineering and its parts Requirements Management and Requirements Development.
- BO04 Communicate basic skills and competences required from a Requirements Engineer.
- BO05 Communicate the meaning of Requirements Engineering for the success of a development and/or maintenance project.
- BO06 Communicate the possible usage of different techniques for requirements elicitation.
- BO07 Communicate the importance of traceability and prioritization and their meaning for the Requirements Engineering processes.
- BO08 Communicate the usage of modeling for creating a business solution for a given business problem.
- BO09 Communicate the importance of Quality Assurance and Quality Control for the Requirements Engineering process.

- BO10 Communicate the similarities and differences in the Requirements Engineering process between common models related to the development/maintenance process.

Level of Detail

The REQB Certified Professional for Requirements Engineering Foundation Level syllabus is intended to support internationally consistent training and examination. This syllabus comprises the following components to reach this goal:

- General instructional objectives describing the intention of the Foundation Level program of REQB certification.
- Learning objectives for each knowledge area describing the objective cognitive learning outcome of the course and the qualifications which the participant is to achieve.
- A list of information to teach, including a description, and references to additional sources such as accepted technical literature, norms or standards, if required.
- A list of terms that participants must be able to recall and understand. The individual terms are described in detail in the REQB "Standard Glossary of Terms used in Requirements Engineering" document.

The syllabus content is not a description of the entire "Requirements Engineering" field of knowledge. It covers the scope and level of detail relevant for Foundation Level certification.

Organization of the Syllabus

There are seven major chapters.

The top-level heading for each chapter shows the topic that is covered within the chapter and specifies the minimum amount of time that an accredited course must spend on the chapter.

Learning objectives to be satisfied by each chapter are listed at the beginning of the chapter.

Within each chapter there are a number of sections. Each section has a minimum amount of time that an accredited course must spend in that section. Subsections that do not have a time associated with them are included within the time for the section.

For example:

1 Introduction to Requirements	90 minutes

Learning Objectives for Foundation Level of Requirements Engineering

The objectives identify what you will be able to demonstrate following the completion of each module.

1.1 Why Requirements are Necessary

LO-1.1.1 Recall the definition of a requirement (K1)

The main header shows that a minimum of 90 minutes must be scheduled to teach the material in the chapter.

Within each chapter there may be several sections each with specific learning objectives. The content of a specific section must allow achieving the relevant learning objectives.

1 Introduction to Requirements	110 minutes

Terms

Business constraint, Business problem, Commitment, Constraint, Criticality, Functional Requirement, Non-functional Requirement, Quality attributes of requirements, Priority, Product, Product Requirements, Requirement, Requirements Development, Requirements Engineering, Requirements Management, Solution, System, Technical constraint, Validation, Verification

Learning Objectives for Foundation Level of Requirements Engineering

The objectives identify what you will be able to demonstrate following the completion of each module.

1.1 The Concept of a Requirement

1.1.1 The Concept of a Problem, a Solution and a Product

LO-1.1.1 Explain the concepts of a problem, a solution and a product (K2)

1.1.2 Definition and Classification of Requirements

LO-1.1.2 Explain the concept of requirements and their classification (i.e. product requirements and constraints) (K2).

LO-1.1.3 Explain the difference between the different levels of requirements (e.g., business requirements, solution (system) requirements and product (component) requirements) and their meaning for Requirements Engineering (K2)

1.1.3 Common Attributes of Requirements

LO-1.1.4 Explain the concept of common attributes of a requirement (e.g., commitment, priority and criticality) (K2)

1.1.4 Quality of Requirements

LO-1.1.5 Understand common problems concerning requirements and explain the role of validation, verification and quality criteria in improving the quality of requirements (K2)

1.1.5 Requirements Engineering, Requirements Management and Requirements Development

LO-1.1.6 Understand the need for process view for Requirement Engineering as well as its impact, to avoid mistakes in Requirement Engineering (K2)

1.2 Standards and Norms

LO-1.2.1 Explain the possible benefits and application of standards for Requirements Engineering (K2)

1.1 The Concept of a Requirement	90 minutes

To understand the discipline of Requirements Engineering and its main inputs and outputs, it is important to draw the context and introduce the most important terms and definitions.

The full list of terms and definitions can be found in the REQB "Standard glossary of terms used in Requirements Engineering".

The purpose of this chapter is to provide basic concepts of Requirements Engineering and how that relates to a requirement itself.

1.1.1 The Concept of a Problem, a Solution and a Product

The main inputs to the Requirements Engineering process are business problems, business needs and/or business objectives. A business problem is the description of what a customer wishes to do in order to realize or improve its business processes. The problem describes or helps to describe the business needs of a customer.

One of the main objectives of Requirements Engineering is to define business solutions for given business problems, business needs or objectives. A solution is the answer to the needs of a customer. Those needs are usually expressed as customer requirements and are inputs to Requirements Engineering. A solution may be a system or a system component, new or changed functionality, changed configuration, process improvement, etc. In general, a solution is a refinement to a higher level requirement.

The output of the realization process is called a product.

For training companies: explain the concept of a problem, a solution and a product with a real-life example of a development or maintenance project.

1.1.2 Definition and Classification of Requirements

Following IEEE 610, a requirement can be defined as:

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

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

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

One of the main purposes of requirements is to express the customer's needs and expectations regarding the planned solution. Requirements also serve as a foundation for assessment, planning, execution and monitoring of project activity and are often a component of service agreements, orders, or contracts. Requirements not only define what is to be done but they establish the solution boundaries, scope of delivery, plan of delivery, and contractual services.

In general, requirements can be classified into product requirements and constraints.

Product requirements describe characteristics, features, qualities and services required from the given product and consist of functional and non-functional requirements. Functional requirements describe the function (behavior) of the product, while non-functional requirements describe so called quality attributes of the products (also referred to as quality goals).

The difference between functional and non-functional requirements can be expressed by the following statements:

- Functional requirements describe *what* the solution does.
- Non-functional requirements describe *how* the solution performs its functions.

Constraints are limitations on the engineering process, a system's operation or its life cycle. There are two types of constraints: business constraints and technical constraints.

- Business constraints express limitations on the project's flexibility to implement the requested solution (for example, financial or time restrictions, limits on the number of resources available, skills of the project team, other organizational restrictions, domain laws, specific standards and regulations).
- Technical constraints are any restrictions that are related to the architecture of the solution (for example, hardware and software platforms, programming language or technology, software that must be used, database size, resource utilization, message size and timing, software size, maximum number of and size of files, records and data elements).

In case of organizations working with a product, a requirement can be external or internal – some requirements may come from the customers and some from the product organization. Requirements coming from different sources should be analyzed together.

Requirements can be classified into types, showing that they deal with different aspects of the product or its development process. Requirements can be classified in abstraction levels representing different levels of detail of a requirement ranging from high-level business needs to

detailed requirements related to software or system functions, or even to elements of the solution design (e.g., screen prototypes, mobile phone keyboards).

Some of the common abstraction levels of requirements include the following:

- Business requirements
    - High level requirements defining what the business wants to achieve but not how to implement it. These requirements express the customer's desires, needs and expectations and they are often referred to as high level business or customer requirements.
- Solution/system requirements
    - Refinement of the customer's requirements describing the solution that is one of several possible ways to fulfill the customer requirements. The solution may contain non-IT related requirements for process changes or organizational/role changes. Solution/system requirements are the expression of the customer requirements in more technical terms that can be used for design decisions.
- Product/component requirements
    - Functions and characteristics of the solution. A complete specification of a product component, including fit, form, function, performance, etc. This abstraction level quite often is missing as it is considered to be a part of the solution design.

When working with requirements on the different levels, it is important to define and maintain traceability (refer to section 4.3 Traceability of Requirements). Traceability is a connection between project artifacts on the different levels, for example, between business requirements and solution requirements or between requirements and associated test cases.

For training companies: provide examples of requirements of different types and levels of abstraction.

1.1.3 Common Attributes of Requirements

The most important attributes for a high-level requirement are the following:

- Commitment
- Priority
- Criticality

Commitment is the degree of obligation for meeting the requirement. It is usually defined through key words allocated to the high level requirements (“shall”, “should”, “may”, “can”) [after ISO/IEC/IEEE 29148:2011, previously IEEE 830]. In some cases, the “must not” key word is used to express aspects that the solution must not do. Key words “should”, “may”, “can” usually relate to business requirements before agreement. After agreeing and baselining, the requirements key words should determine the degree of obligation in more strict terms: “will” or “shall”.

The level of obligation of a requirement can be also expresses using MoSCoW notation (Must have, Should have, Could have, Will not have this time but will have in the future).

Once the solution provider and the customer reach an agreement, commitment to the requirements is obtained from project participants.

Requirements may evolve throughout the project. As requirements evolve, obtaining commitment from the project participants for the changed requirements ensures commitment to the resulting changes in project plans, activities, and work products.

One of the most important implications of a commitment is responsibility for the final products. There may be legal responsibilities related to the quality of the product. Legal responsibilities are often related to a specific requirement (for example, an environmental requirement for a nuclear plant or a safety requirement for an airplane) that must be satisfied in the delivered product. The responsibilities may also relate to defects in the product. Commitment is one way of expressing fulfillment of legal requirements.

Legal responsibilities should be defined in the contract between the supplier and the customer. Some industries may also be required to meet contractual or legal requirements, or industry-specific standards.

The next requirement attribute – priority – expresses the importance/urgency of a requirement. According to [SWEBOK], in general, the higher the priority, the more essential the requirement is for meeting the overall goals of the product. Usually classified on a fixed-point scale such as mandatory, highly desirable, desirable, or optional, the priority often has to be balanced against the cost of development and implementation.

Criticality of a requirement is the result of an evaluation of the risk of the damage that would occur if the requirement were not fulfilled. . The criticality is expressed in levels; the higher the level, the more severe the consequences in case of a functional failure.

In some cases, attributes of requirements can be also expressed in a form of the Kano model. This model focuses on differentiating product features in the following way:

- Threshold or Basic attributes – these are the features that the product must have in order to meet customer needs.

- Performance attributes – a skill, knowledge, ability, or behavioral characteristic that is associated with the way in which the product will deliver its capabilities to the user. Performance attributes improve customer satisfaction.
- Excitement attributes – attributes unforeseen by the customer (sometimes called "unknown needs") that are to be discovered by the supplier. They enlighten the consumer and can provide a significant competitive advantage.

The Kano model can be used to express both priority and commitment.

In addition to the attributes described above, requirements may be categorized by the following [ISO/IEC/IEEE 29148:2011, previously IEEE 1233]:

- Identification
- Feasibility
- Risk
- Source
- Type

For training companies: explain the common attributes of a requirement and provide some examples demonstrating their meaning.

1.1.4 Quality of Requirements

Requirements Engineering is one of the most important success factors for any project. Requirements are the basis for further work; therefore, it is very important to ensure the best possible quality of requirements and other products of the Requirements Engineering process. The most important issues to be considered when working with requirements are the following:

- Unclear business objectives for the requirements or the project itself
- Communication problems (often resulting from language and/or knowledge barriers, including lack of business domain knowledge), Vague formulations (often resulting from an insufficient or unmeasurable definition of a requirement, or lack of common glossary)
- Volatility of the requirements (often caused by unclear business objectives for the requirements or the project itself)
- Bad quality of the requirements (see quality criteria for requirements below)
- Gold plating (adding extra features that weren't really needed)

- Insufficient stakeholder involvement
- Missing stakeholders (often resulting from a poor Requirements Engineering process)
- Inaccurate planning (often caused by unclear or missing business objectives for the requirements or the project itself)
- Insufficient requirements or solution specifications (resulting in gaps and missing or fragmented requirements)

Some of the problems listed above can be resolved quite simply by applying quality criteria for the requirements. According to [Wiegers], common quality criteria for requirements define that each requirement must be:

- Correct – the requirement must accurately describe the features to be provided. The point of reference to evaluate the correctness is the source of the requirement (for example, customers or a higher-level system requirement).
- Feasible – the requirement must be possible to implement within the known capabilities and/or limitations of the system and the environment.
- Necessary – the requirement should document what the customer (or other stakeholders) really need and what is required in order to accomplish an external requirement or interface or a specified standard.
- Prioritized – the requirement should have a priority assigned indicating how essential it is for a particular product release.
- Unambiguous – the requirement should be interpreted only in one way. Different readers of a requirement should have the same interpretation and understanding of a requirement.
- Verifiable – the requirement should be possible to verify if it is implemented correctly.
- Singular – the requirement does not contain multiple requirements; this implies a sufficient granularity to specify a single requirement.
- Design (implementation) independent – the requirement should describe “what must be done”, not “how to do it”. The content of a requirement should not prescribe or imply its implementation details.

Quality criteria can be applied not only to a single requirement, but can be used for a requirements specification as well.

The major quality criteria for specifications state that the requirements specification must be:

- Complete – no requirements or necessary information should be missing in the requirements specification. Completeness is also expressed as a desired characteristic of an individual requirement and the level of detail.
- Consistent –requirements described in the specification cannot conflict with other product requirements or with higher-level (system or business) requirements.
- Modifiable – the specification must allow introducing changes in the requirements. A history of changes made to each requirement should be maintained.
- Traceable – each requirement should be possible to link to its source (for example, a higher-level system requirement, a use case, or a customer's statement) and related implementation artifacts (for example, design elements, source code, and test cases).

For training companies: explain the common quality criteria of requirements and provide some examples of requirements meeting and not meeting the quality criteria.

Assuring the required level of quality for requirements or other products of the Requirements Engineering process can be supported by validation and verification.

According to CMMI, validation activities demonstrate that a product or product component fulfills its intended use when placed in its intended environment. So, you know that "you built the right thing". Customers often identify product descriptions or requirements in a weak manner and validation helps to understand what is needed (by using tools such as scenarios, use cases, prototyping, etc.).

Validation is usually conducted with the support of a customer or stakeholders at the customer site and aims to confirm that the requirements or the requirements specification accurately describes what the customer needs.

According to CMMI, verification provides checkpoints at which selected deliverables or intermediate products are verified to confirm that they meet their requirements. Verification activities focus on incremental confirmation of the implementation of the requirements; they enable early and ongoing confirmation of building the product right.

Verification is a comparison of an intermediate product with its specifications. It is thereby determined if the product was developed correctly and if the specifications that were determined during the previous phase were fulfilled. The most common techniques for verification are reviews, static analysis and dynamic testing. For validation, reviews and dynamic testing are used.

The difference between validation and verification can be expressed by the following:

- Verification - did we create the product correctly?
- Validation - did we create the correct product?

1.1.5 Requirements Engineering, Requirements Management and Requirements Development

Requirements Engineering (RE) can be defined as a sub-discipline of System Engineering, focused on determining, developing and managing the requirements of hardware and software systems [SWEBOK, Sommerville]. According to CMMI, Requirements Engineering encompasses Requirements Management and Requirements Development.

The purpose of Requirements Management is to manage baselines of an agreed set of requirements of the solution and to ensure alignment between those requirements and the project's plans and work products.

Requirements Management constitutes both a working framework for Requirements Engineering as well as supporting processes for the Requirements Development process. The other role of Requirements Management is to establish and provide interfaces to the other development and managerial processes interfacing with Requirements Engineering such as Project Management, Risk Management, Design, Configuration Management, Change Management and Quality Assurance.

Requirements Development is a collection of activities, tasks, techniques and tools to identify, analyze, document and validate requirements on the different abstraction levels. It includes the process of transforming needs into requirements as well as development (refinement of the high level requirements) of a solution for these requirements.

Good Requirements Engineering process is one of the major success factors for any project. Well elicited, analyzed, documented and managed requirements lower project risks as they form a clear and understood basis for the solution design. However, the meaning of good requirements and Requirements Engineering process is still not well understood, or is simply neglected.

The reasons for neglecting Requirements Engineering may be the following:

- High time and cost pressure
- An exclusive orientation toward fast results
- Consideration of only the functional requirements
- Lack of understanding of the importance of Requirements Engineering for the success of a project

Consequences of a poor Requirements Engineering process may be the following:

- Low quality requirements
- Requirements that often change during the product development
- Requirements that do not fulfill the Acceptance Criteria and/or do not provide a value to the customer or other stakeholders
- Missing requirements or constraints

1.2 Standards and Norms	20 minutes

Note: It is not required that specific standard identification numbers be memorized.

There are many standards and process norms that are useful for Requirements Engineering. Some of them provide process models related to solution development, others provide guidelines for writing different requirements specifications or for requirements classification.

One of the main purposes of standards is to have a national and international alignment of products and processes. Standards normalize development methods and deliverables, provide common terminology and facilitate communication in business and technology.

The most important standards and norms that are used in the REQB scheme are listed below.

Standards:

- ISO 9000. The ISO 9000 family of standards provides requirements for a quality management system and defines concepts and basics of a Quality Management System (QMS).
- ISO/IEC 25000 (previously ISO 9126). It defines a quality model with six main categories (functionality, reliability, usability, efficiency, maintainability, portability) which can be a basis for Requirements Elicitation, Specification or Validation and Verification.
- IEEE Standard 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology
- IEEE Standard 830-1998. IEEE Recommended Practice for Software Requirements Specifications (also known as Recommended Practice for SRS).
- IEEE Standard 1233-1998. IEEE Guide for Developing System Requirements Specifications (also known as Guide for Developing of SyRS).
- IEEE Standard 1362-1998. IEEE Guide for Information Technology-System Definition – Concept of Operations (ConOps) Document
- Note: IEEE 830, IEEE 1233 and IEEE 1362 have been replaced by ISO/IEC/IEEE 29148:2011
- SWEBOK - The Guide to the Software Engineering Body of Knowledge (known as ISO Technical Report 19759). SWEBOK describes generally accepted knowledge about software engineering. Its 10 knowledge areas summarize basic concepts and include a reference list pointing to the detailed information. One of the areas is dedicated to Requirements Engineering.
- SEBOK – The Guide to the System Engineering Body of Knowledge.

Process norms for development:

- ISO 12207. Standard for Software Life Cycle Process.
- ISO 15288. System Life Cycle Process.

Both can be used to support organization of the solution development process.

Process norms for process evaluation and improvement:

- ISO 15504 Information Technology — Process assessment, also known as SPICE (Software Process Improvement and Capability Determination).
- CMMI. Capability Maturity Model Integrated.

Both support process improvement and can be used for process evaluation and improvement. They define key areas for Requirements Engineering.

For training companies: explain why these standards are relevant and important when defining a Requirements Engineering process.

2 Context of Requirements Engineering	40 minutes

Terms

Business Analysis

Learning Objectives for Foundation Level of Requirements Engineering

The objectives identify what you will be able to demonstrate following the completion of each module.

2.1 Requirements Engineering in Context

LO-2.1.1 Explain the context of Requirements Engineering and how Requirements Engineering is a part of solution development and the maintenance life cycle (K2)

2.2 Connected Processes

LO-2.2.1 Explain the relationships and interactions between Requirements Engineering and other disciplines in the development project (K2)

2.1 Requirements Engineering in Context	20 minutes

Requirements Engineering is not performed in isolation. It is linked with other disciplines (Figure 1) and should be incorporated into the overall solution development process.

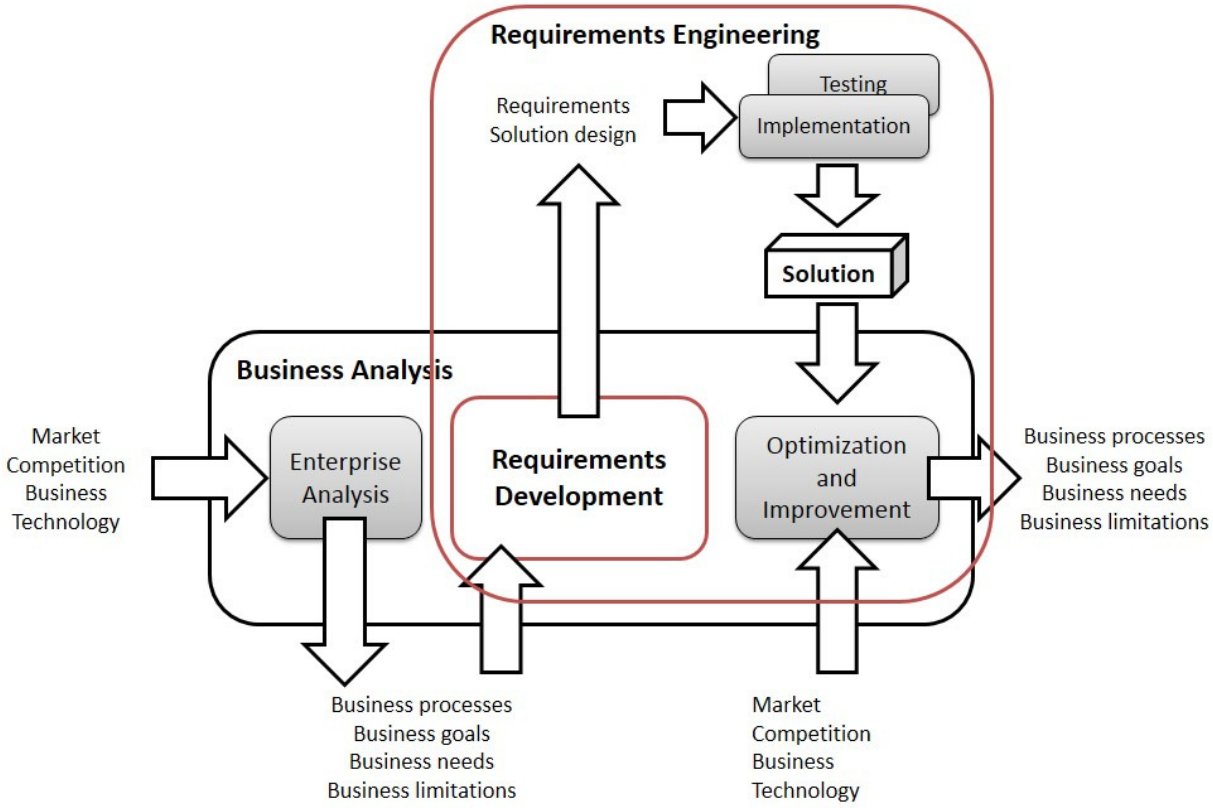

Figure 1 The context of Requirements Engineering

The starting point for Requirements Engineering is Business Analysis. In order to propose the best solution for a given business problem, it is important to define the problem correctly [IBAQB].

Business Analysis is a discipline providing a set of activities, tools and methods aiming to establish business needs, problems and goals, determine proper solutions to satisfy those needs and resolve given business problems. These business solutions may include development of software systems or software components, extensions of existing software, improvements to the business process, changes to the organization, etc. Business Analysis defines the business problem to be solved by a

specific solution which becomes an input for further Requirements Engineering. The business goals and needs established during Business Analysis are developed into requirements for the solution. Therefore Requirements Engineering can be seen as a continuation or a part of the Business Analysis process. The output from Requirements Engineering is the solution design, further implemented, tested and finally provided to the customer.

The implemented solution rarely stays unchanged for a long period of time. The market and business change, technology is evolving all the time, and new connections within the business appear. Nearly all systems and solutions will be changed and see release of new versions. It is important that both Business Analysis and Requirement Engineering are handling the maintenance activities too. That is why Business Analysis – and Requirements Engineering – continues during the whole life cycle of the solution. After delivering the solution Business Analysis seeks for improvements that will increase the market, improve the efficiency of operating or gain competitive advantage. New business needs will appear. Those business needs are developed by Requirements Engineering to propose new, improved solutions or extend the existing solution with new, better features.

For training companies: explain the context of Requirements Engineering in real-life examples. Provide examples of relationships between Requirements Engineering and Business Analysis (in terms of different products and activities).

2.2 Connected Processes	20 minutes

Requirements Engineering, and especially its part called Requirements Management, operates in a larger context and has strong relations with the following processes:

- Business Analysis – As explained in the previous section, Business Analysis and Requirements Engineering are strongly connected.
- Project Management – The project manager should include requirements specific activities in his project management plan, allocate the required time and resources and ensure they are correctly carried out. Requirements Engineers may have to produce a Requirements Plan which details the tasks required to complete the Requirements Specification.
- Risk Management – Some requirements may introduce project or product risks which should be managed via the Risk Management process.
- Analysis and Design – Requirements are a mandatory input for the analysis and design activities. Traceability between requirements and analysis and design items is required.
- Configuration and Change Management – Requirements should be configuration managed and follow the Configuration Management process. A requirement may be associated with an item which is configuration managed.
- Testing – Testing provides feedback on the differences between the stated requirements and the developed functional and/or non-functional attributes of the product. Traceability between requirements, tests and other artifacts is mandatory in order to provide complete visibility regarding test coverage. Depending on the associated test items, a specific requirement may have a status showing whether or not it is implemented.
- Release Management – The starting point for Release Management is the result of Requirements Management. For each solution or product released, it must be possible to identify the requirements implemented in that release.

With different developments models, or different products, Requirements Engineering may be connected to other processes as well.

For training companies: provide examples of relationships between Requirements Engineering and other disciplines (in terms of different products and activities).

3 Requirements Engineering Process	80 minutes

Terms

Customer, Generic Requirements Engineering Process, Requirements Developer, Requirements Engineer, Requirements Manager, Supplier, Stakeholder

Learning Objectives for Foundation Level of Requirements Engineering

The objectives identify what you will be able to demonstrate following the completion of each module.

3.2 Generic Requirements Engineering Process

LO-3.2.1 Explain the common objectives of Requirements Engineering and the typical activities of the Requirements Engineering process (K2)

3.3 Roles and Responsibilities

LO-3.3.1 Explain the roles of Requirements Development and Requirements Management (K2)

LO-3.3.2 Recall the basic roles in Requirements Engineering and explain their effect on the Requirements Engineering process (K1)

LO-3.3.3 Recall the main skills and competencies required from a person involved in Requirements Engineering (K1)

3.1 Introduction to the Requirements Engineering Process	40 minutes

The structure of the Requirements Engineering process depends on different factors, such as the organization's culture and maturity, the development model used, etc. REQB uses the following categorization:

- Generic Requirements Engineering process
- Requirements Engineering process in development and maintenance models and approaches
- Requirements Engineering process in maturity models

The Generic Requirements Engineering process should be the starting point for any organization involved in solution development work as it provides the most crucial processes for handling requirements. This generic Requirements Engineering process can be adapted to the specific needs of an organization and the development or maintenance model that is to be applied. The result of this adaptation is the Requirements Engineering process in development and maintenance models and approaches. In addition to development process models, maturity models exist. Those maturity models are mostly used to evaluate the current maturity of a specific organization and introduce improvements needed to increase the efficiency of the organization. Therefore, we can define another variant of the generic Requirements Engineering process – Requirements Engineering process in maturity models. This variant of the Requirements Engineering process is covered in the REQB Advanced Level program.

The generic Requirements Engineering process should be adapted to the needs of the specific organization, taking into account the type of solutions being developed, business domain, business processes, organizational culture, and the level of competence and ability of the personnel in charge of the Requirements Engineering process.

For training companies: explain the REQB approach to Requirements Engineering process and demonstrate the need for adjusting the generic Requirements Engineering process for specific needs.

3.2 Generic Requirements Engineering Process — 20 minutes

Requirements Engineering is a discipline that includes processes for the elicitation, structuring and managing of requirements. Specific activities covered by the generic Requirements Engineering process include the following:

- Requirements Elicitation
- Requirements Analysis
- Requirements Specification
- Requirements Validation and Verification
- Requirements Traceability
- Configuration and Change Management
- Quality Assurance

The Requirements Engineering process is a structured set of the activities mentioned above. The activities are organized under a Requirements Management process or a Requirements Development process, depending on the purpose and the phase of the solution development. A complete process description should include connections with other related disciplines and areas (for example, Business Analysis or testing), input to and output from the activities, information about the responsibility for a specific activity and its outputs, required competencies and the tools to support the activities (more details in Chapter 7 Tool Support).

The main activities, inputs and outputs to the specific Requirements Engineering processes are presented in Table 2.

Process	Input	Main Activities	Output
Requirements Development			
Requirements Elicitation	Business needs Initial scope definition Stakeholders	Collecting requirements from stakeholders Detailing known high-level requirements Excluding unnecessary features	Business Requirements Limitations Assumptions
Requirements Analysis	Business Requirements Limitations Assumptions Stakeholders Organizational assets Modeling techniques Tools	Elaborating business requirements into system/solution requirements and down to product-component requirements Resolving conflicts between requirements Establishing the boundaries for the solution Developing models of the business solution	Agreed requirements Scope definition Business solution model
Requirements Specification	Requirements Limitations Assumptions Stakeholders Organizational assets	Documenting requirements	Requirements specification documents
Requirements Validation and Verification	Requirements Requirements specification documents	Checking the quality of requirements and requirements specification documents	Validated requirements Validated requirements specification documents Defects Issues Changes

Process	Input	Main Activities	Output
Requirements Management			
Traceability of Requirements	Requirements Stakeholders Organizational assets Tools	Establishing a structure for traceability Defining and maintaining traceability of requirements	Traceability matrix
Configuration and Change Management	Requirements Tools Change requests Project plans Traceability matrix	Identifying and managing configuration items Requesting, determining attainability, planning, implementing, and evaluating changes to a system, documents or other products of the project Performing risk analysis Processing changes and ensuring traceability of changes	Baselined requirements Risk analysis results Changed requirements and requirements specification documents Project plans
Quality Assurance	Requirements Requirements specification documents Solution models Stakeholders Organizational assets Standards Tools Traceability matrix Risk analysis	Ensuring that the different Requirements Engineering processes and their products are of good quality Applying relevant standards Performing reviews and other QA activities	Requirements Requirements specification documents Business solution models Defects Issues Changes Improvements Requirements Engineering processes and activities defined and trained

Table 2 Overview of the Requirements Engineering activities, inputs and outputs

The efficiency of the Requirements Engineering process is not the same in all organizations or projects. There are some factors, external or internal, that can have a negative influence on the Requirements Engineering process or products.

Factors from the internal side (i.e., inside the product supplier's organization) include:

- Lack of knowledge of the business or user's domain
- Lack of knowledge of technologies requested by the customer
- Ineffective Requirements Engineering approach/methodology/tools
- Insufficient personnel experience, skills and competency

Factors from the external side (i.e., outside the product supplier's organization) include:

- Lack of communication or ineffective approach to communication of the requirements, needs, or expectations
- Unclear and/or changing business objectives resulting in unstable requirements
- Insufficient knowledge about the product development process
- Insufficient involvement of users and/or business stakeholders

Requirements Engineering deals with many tasks, depending on its role in the solution development or maintenance life cycle as well as constraints and characteristics of the planned solution.

The Requirements Engineering process and its products (such as requirements specifications or a business solution proposal) should provide the best value for the customer and other stakeholders. To be able to ensure that the planned solution best suits the customer's needs and meets expectations, requirements should be developed from the customer's point of view. Sample methods of the customer-oriented Requirements Engineering process are the following:

- Customer-oriented analysis and design
- Prototyping approach
- Using demonstrations as a means to validate the increments of the system

For training companies: provide a sample overview of the Requirements Engineering activities, inputs and outputs for a specific business area/project.

3.3 Roles and Responsibilities	20 minutes

A successful Requirements Engineering process requires clear roles and responsibilities definitions. In general, the roles can be classified as roles that affect or are affected by Requirements Engineering and roles within Requirements Engineering.

The most important roles affecting or affected by Requirements Engineering are the customer and the supplier. The customer is a person, group or organization requesting the solution, while the supplier (also called a vendor) is a person, group or organization providing the solution. The customer formulates his needs and provides the initial business needs and expectations. These are usually provided together with the offer/service request. The supplier is responsible for exploring these needs and extracting requirements on that basis. The main goal of the supplier is to deliver solutions to fulfill the client's needs.

The roles within Requirements Engineering can be defined as Requirements Manager and Requirements Developer. Both roles also can be called Requirements Engineers.

The large variety of tasks related to Requirements Engineering has resulted in many different titles for people performing Requirements Engineering oriented tasks. Different organizations use their own names, such as Requirements Engineer, Requirements Manager, Requirements Developer, Business Analyst, System Analyst, Solution Architect, System Architect, Designer, etc. Regional variants of these roles exist as well depending on culture, habits and traditions. The problem is that specific responsibilities and competencies needed for the roles mentioned above are rarely defined, clear or understood.

To provide a common understanding of the responsibilities and roles, REQB defines the roles of Requirements Manager and Requirements Developer. This classification results from the difference in the tasks performed in the Requirements Management and Requirements Development activities. A Requirements Manager is a person with overall responsibility for documenting, analyzing, tracing, prioritizing and coordinating the agreement to the requirements and then controlling changes and communicating requirements to relevant stakeholders. A Requirements Developer is a person mainly involved in the elicitation, analysis, documentation and prioritization of requirements.

A Requirements Manager and a Requirements Developer should possess the following skills:

- Methodological competence (i.e., practical knowledge of the Requirements Engineering process, methods, techniques and tools)

- Knowledge about sources of best practices and the most important standards related to Requirements Engineering
- Precise, analytical and clear thinking
- Ability to act in a structured way
- Moderation and negotiation skills
- Self-confident manner
- Ability to argue and convince
- Language and communication skills
- Stress resistance

Another role associated with Requirements Engineering is a stakeholder. A stakeholder is a group or individual that is affected by or is in some way accountable for the outcome of an undertaking. Stakeholders are individuals and organizations that are actively involved in the project, or whose interests may be affected as a result of project execution or project completion. They may be either natural persons, legal entities or abstract persons.

Stakeholders often have conflicts of interest among each other, resulting in contradictory requirements. The problem of conflicting requirements needs to be addressed during the Requirements Analysis activity.

There may be many categories of stakeholders including:

- Customers
- End-users
- Project Managers
- Engineers responsible for system development and maintenance
- Customers of the organization who will use the system
- External bodies (e.g., regulators)
- Domain experts (i.e., SMEs)
- Suppliers of the related solutions
- Maintenance organizations

Identification of all stakeholders is necessary to understand all points of view regarding the planned solution. Missed stakeholders can result in missing requirements or important limitations affecting the scope or shape of the solution.

For training companies: describe a typical stakeholder, their points of view on the project and its deliverables, and main goals and communication patterns (e.g., Business Owner/Sponsor, Project Manager, Customer, Requirements Engineer, Agile development team, Scrum Master).

4 Requirements Management	160 minutes

Terms

Baseline, Change, Change Control Board, Change Management, Change Request, Configuration Management, Configuration Item, Horizontal Traceability, Impact Analysis, Metric, Quality Assurance, Quality Control, Product Risk, Project Risk, Review, Risk, Risk Analysis, Risk Identification, Risk Management, Risk Management Plan, Risk Mitigation, Traceability, Vertical Traceability

Learning Objectives for Foundation Level of Requirements Engineering

The objectives identify what you will be able to demonstrate following the completion of each module.

4.2 Project and Risk Management

LO-4.2.1 Explain why Requirements Engineering is important in projects and provide proper examples (K2)

LO-4.2.2 Recall the errors that can occur in Requirements Engineering (K1)

LO-4.2.3 Distinguish between typical project and product risks related to requirements (K2)

LO-4.2.4 Describe, using examples, how Risk Analysis and Risk Management may be used for Requirements Management (including requirements planning) (K2)

4.3 Traceability of Requirements

LO-4.3.1 Understand the purpose of traceability (K2)

LO-4.3.2 Recognize the different kinds of traceability (K1)

4.4 Configuration and Change Management

LO-4.4.1 Explain the characteristics of Configuration Management and Change Management and the role of a Change Control Board (K2)

LO-4.4.2 Explain the terms: configuration item, defect and change request (K2)

4.5 Quality Assurance

LO-4.5.1 Recall the factors that influence the quality of the Requirements Engineering process and products (K1)

LO-4.5.2 Explain how products of Requirements Engineering support testing (K2)

LO-4.5.3 Explain how metrics can be used for evaluating and improving the quality of the Requirements Engineering process and its deliverables (K2)

4.1 Introduction to Requirements Management	10 minutes

Requirements Management is mainly a collection of managing and supporting activities which are needed to assure that the Requirements Development process is executed properly during the product life cycle. Configuration and Change Management as well as solution maintenance are also activities within Requirements Management.

Generally Requirements Management deals with the following activities:

- Traceability of Requirements
- Configuration and Change Management
- Quality Assurance

4.2 Project and Risk Management	40 minutes

Some of main reasons why projects fail are related to requirements. Neglecting Requirements Engineering can result in requirements are not precise, are contradictory, do not fulfill the criteria and do not satisfy stakeholders' needs. Careful and structured Requirements Engineering is a necessary part of any project.

To minimize the project risks, Requirements Engineering should contribute to the following areas:

- Project conception – Requirements Engineering supports identifying customers, stakeholders, objectives and visions, needs and expectations regarding the solution of the problem and supports establishing high-level requirements.
- Contract negotiations – Requirements Engineering supports evaluating customer's requirements, and determining the initial scope, required resources for the project and determining the costs of development (i.e., implementation of requirements).
- Project definition – Requirements Engineering includes the definition of roles, tasks, activities and additional processes (e.g., Change Management). It also contributes to the initial architecture design and test process.
- Project execution – at this step Requirements Engineering provides a base for requirements development and requirements verification and validation (i.e., testing). It should also monitor changes and require reviewing plans and adjusting them to the current scope of the solution when the requirements change.

Ineffective or poor Requirements Engineering increases the project risks by allowing errors to propagate to subsequent steps in the solution development. Therefore, when planning the Requirements Engineering process, both best practices and common risks and errors should be taken into account. The most common problems related to Requirements Engineering are the following:

- Unclear business requirements and a lack of business goals to be achieved within the solution
- Changing requirements often resulting from the unclear objectives of the project and lack of a clear definition of the customer's business domain. Changes to requirements are not perceived as an error in Agile and iterative approaches.
- Unclear responsibilities (on both the customer's and the supplier's side)
- Gaps between stakeholders expectations and the project's deliverables

- Insufficient stakeholder involvement
- Lack of traceability, often resulting in imprecise estimates of the impact of requirement changes on the other areas of the product under development
- Imprecise expense or scope estimates, and/or project definition with milestones that cannot be achieved (often resulting from unclear requirements)

The potential for the problems listed above can be perceived as a risk. To deal with those risks – and other risks as well – a Risk Management process is necessary.

Efficient Risk Management is a key to lowering project and product risks. Identification, proper analysis, and planning adequate responses to risks help to minimize the chances of a risk being realized and the resulting consequences.

Risks are not isolated to requirements. In general, risk can be defined as the chance of an event or situation occurring and resulting in undesirable consequences or a potential problem. Risk is expressed in levels, where the level of risk is determined by the likelihood of an adverse event happening and the impact (the harm resulting from that event) [ISTQB].

There are two main types of risk: product risk and project risk.

Project risks are the risks that surround the project's capability to deliver its objectives, such as:

- Organizational factors (e.g., skills, training and staff shortages, political issues, problems with stakeholders communicating their needs and expectations regarding the planned solution, or improper attitude toward or expectations of Requirements Engineering)
- Technical issues (e.g., problems in defining the right requirements or the technologies/architecture solutions, the extent to which requirements cannot be met given existing constraints, or low quality of the design, code, configuration data, tests, requirements or solution specifications)
- Supplier issues (e.g., components developed by a third party not delivered on time, or contractual issues)
- Business risk based on poor quality (e.g., the risk of losing clients if the delivered solution is found to be unreliable or inefficient)

Product risks are potential failure areas (adverse future events or hazards) in the software or system. These are risks to the quality of the product. These include:

- Higher risk of failure of the delivered product (software or system unable to perform a required function within specified limits)
- Low quality of the solution documentation (e.g., user manuals, installation guides)
- The potential that the software/hardware could cause harm to an individual or company

- Poor product characteristics (e.g., functionality, reliability, usability, performance)
- Poor data integrity and quality (e.g., data migration issues, data conversion problems, data transport problems, violation of data standards)
- A product that does not perform its intended functions and does not satisfy the needs of stakeholders

The process of Risk Management supports identifying potential factors that may have a negative effect on the execution of a project and preparing proper actions to deal with the risk if it is realized.

Risk Management consists of the following activities [ISTQB]:

- Risk Identification
- Risk Analysis (Assessment)
- Risk Mitigation

When identifying risks it is important to explore the whole network of stakeholders, as different groups of stakeholders or individual stakeholders may recognize different risks. For example, a Business Sponsor may be concerned with the value provided by the solution, while the most important risk for a Requirements Engineer might be a lack of communication with the customer's business representatives.

Risk Analysis includes establishing the risk levels – likelihood and potential impact. Both attributes can be expresses in levels, or – more precisely – in a form of numerical estimations (for example, likelihood of an adverse event happening can be expresses as a percentage value, while the impact can be expressed as a financial loss).

Risk Mitigation plans appropriate reactions for risks identified as most crucial based on the results from the risk analysis. Basic techniques to mitigate the risk can be divided into four major categories:

- Risk avoidance
- Risk reduction
- Risk sharing
- Risk retention

When working on the Risk Management process, it is important to establish and maintain a Risk Management Plan. This plan should be created before creating the Project Plan and then periodically updated (for example, after each iteration or milestone). The Risk Management Plan should provide effective security controls for managing the risks and should contain a schedule for control implementation and a list of responsible persons.

The Risk Management Plan usually includes:

- List of risks (classified into types)
- Owner
- Probability (likelihood) of occurrence for each risk
- Severity of impact for each risk (including cost, if applicable)
- Mitigation strategies for each risk (including the person/group responsible for taking the risk mitigation actions)
- Traceability of risks to requirements and other project artifacts
- Risk assessment matrix

For training companies: provide examples of risks related to requirements and sample actions to manage the risks.

4.3 Traceability of Requirements	40 minutes

Requirements are not stable, but continue to evolve during the project life cycle. Reasons for continued evolution and proposed changes include the following:

- New insights or customer needs (e.g., resulting from new regulations, changes in the business, new products)
- Continued work (e.g., detailing already defined higher level requirements, next project phase, improving and optimizing already implemented features)
- New connections within the project (e.g., integration with new systems, new access channels such as the Internet or mobile channels)

Depending on the degree of Requirements Analysis and/or implementation, a different status will be assigned to the requirement. The life cycle of a requirement is often expressed via different statuses, for example:

- New (proposed)
- Approved
- Conflicted
- Implemented
- Modified
- Deleted
- Deployed

Different approaches and organizations may use different requirement life cycles (and different statuses). In many cases, life cycles of requirements, change requests and defects are very similar and managed using the same tool.

Traceability provides a method for managing evolving requirements and other artifacts related to those requirements.

Traceability provides a check that all important steps of the development process have been carried out. It should be implemented bi-directionally for all artifacts (for example, from requirements to design artifacts and from design artifacts to requirements). Traceability is also the basis for testing, verification and validation.

The common goals of traceability are the following:

- Impact analysis
- Coverage analysis
- Proof of implementation
- Use of the requirement (requirements tracking as a proof that the requirements are being used and how they are used)
- Reusability of requirements

In order to ensure good traceability, it is important to label the requirements uniquely.

There are two basic types of traceability:

- Horizontal traceability
    - Presents dependencies between different types of artifacts on the same level of abstraction
- Vertical traceability
    - Presents dependences between artifacts on different levels of abstraction

In the context of requirements, horizontal tracing is usually used to present relationships between different types of requirements (for example, horizontal tracing between a software functional requirement and a GUI requirement supplying the specific function). Vertical tracing is often used to manage coverage of requirements on the different abstraction level (for example, vertical tracing between business requirements and solution requirements).

Both types of tracing can be used to support impact analysis.

According to ISO/IEC/IEEE 29148:2011 (previously IEEE 830), in static templates tracing can be implemented by:

- Textual references in requirements to the identifiers of other artifacts
- Textual references in attributes
- Hyperlinks
- Matrices

For training companies: explain the concept of traceability and provide examples of its application.

4.4 Configuration and Change Management	30 minutes

To ensure proper Requirements Management, a Configuration Management process must be implemented. In many cases requirements are not stable and the subsequent changes may affect other related project artifacts.

The purpose of Configuration Management is to establish and maintain the integrity of the products (components, data, and documentation) and the software artifacts, throughout the project and product life cycle.

Configuration Management is a discipline that applies technical and administrative tools and techniques to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics and record and report change processing and implementation status, and finally – verify compliance with specified requirements [IEEE 610]

A Configuration Item is an artifact, document, product (hardware and/or software) that has an end user purpose and is treated as a single entity in the Configuration Management Process [IEEE 610].

Common configuration items used in Requirements Engineering include:

- Individual requirements
- Requirements specifications
- Models

Configuration Management ensures that all work products (outcomes) of Requirements Engineering are identified, version controlled, tracked for changes, related to each other, and related to other projects items (e.g., development and test artifacts) so that traceability can be maintained throughout the production process.

When planning the project, Configuration Management procedures and infrastructure (tools) should be chosen, documented and implemented. This is due to the fact that configuration items should be defined and brought under change control as early as possible.

For training companies: provide examples of Configuration Management activities for different work products of the Requirements Engineering process.

An important aspect related to configuration is a change. Changes of the requirements may be requested at any time during the realization of the project and after releasing the completed product to the production environment.

The source of a change may be the following:

- Extension or change of existing functionality
- Request for new functionality/requirement
- Defects found in the product/documentation resulting in a need for a change
- Changes resulting from external factors (e.g., organizational changes, regulatory changes)

Changes usually happen and it is important to plan changes in terms of process and time.

It is important to distinguish between a change and a defect. A defect is a deviation from the required state of the system. A change is a modification or addition of features, requirements or functions.

Changes are processed via the Change Management process. The Change Management process is the process of requesting the change, determining attainability, planning, implementing, and evaluating changes to the software/system, documents or other products of the project. The purpose of Change Management is to support the processing of changes and ensure traceability of changes.

The Change Management process usually includes the following activities:

1. Identifying a potential change
2. Requesting new/modified functionality (submitting a Change Request)
3. Analyzing the change request (usually done by a Change Control Board and including Impact Analysis of the change)
4. Evaluating the change (including assessing the risk, cost, and effort of the change in order to make a decision whether or not to implement the change)
5. Planning the change (if the change has been approved for implementation)
6. Implementing the change
7. Reviewing and closing the change
8. Rolling out the change into the test/production environments

A change should be submitted as a formal Change Request (CR) document (sometimes called a Request for Change (RFC)). This document includes the following information:

- The name of person/department or other entity requesting the change
- Submission date
- The reason for the change

- A description of the requested solution
- Planned (desired) date of implementation of the change (if applicable)
- Planned (or allowed) budget for the change (if applicable)

Changes are usually checked and decided on by a Change Control Board (CCB) or Configuration Control Board. A Change Control Board is a committee that makes decisions regarding whether or not proposed changes should be implemented based on the provided information (such as the risk related to the change, its impact, and effort required for implementation). The CCB is composed of project stakeholders or their representatives. In Agile development models, the role of the CCB is usually performed by the development team together with the Product Owner and customer representatives.

Introducing a CCB supports a controlled way to review, assess and implement a change to a product or service.

The Change Control Board usually consists of the following roles:

- Project management
- Requirements Engineers
- Development management
- Quality Assurance (Quality management, Test management)
- Business management
- Customer and/or end user representatives

Depending on its complexity and impact, a change may have various effects on the system. Small changes may require minor modifications, while complex ones may radically change the logic of the system. Each change should be carefully analyzed in order to identify related risks and evaluate the value of modification against the predicted risks.

When deciding on changes it is important to analyze the impact of the change on the project. This is usually done via Impact Analysis. Impact Analysis uses traceability to check which artifacts will need to be changed or at least should be checked when introducing a specific change.

Changes in requirements may have various effects on the project. Most common effects are the following:

- Works resulting from the change (depending on the phase of the project):
    - Updating analysis and design artifacts (e.g., requirements or solutions specifications)

- Updating technical and user documentation
- Changing the test strategy, test plans and test cases
- Updating training needs/plan
- Extending/shortening the programming work scope
- Changing testing preparation and execution scope

- Changes to the schedule, budget, and resources

Implemented changes should be verified before being deployed into the test/production environments.

4.5 Quality Assurance	40 minutes

When defining the Requirements Management process, it is also necessary to define necessary Quality Assurance (QA) activities to be applied in order to ensure that the different Requirements Engineering processes and their products are of good quality.

Quality Assurance is defined as “all the planned and systematic activities implemented within the quality system, and demonstrated as needed, to provide adequate confidence that an entity will fulfill requirements for quality” [ISO 9000]. This definition implies that the actions taken are “planned and systematic” and they “provide adequate confidence” that the desired level of quality will be achieved. These actions include operational techniques and activities used to fulfill the requirements for quality.

To achieve the required level of quality, Quality Control is needed as well. The main goal of Quality Control is to steer and control the quality of products or services through use of operative methods so that they meet specified standards of quality. These operative methods involved in Requirement Engineering include Project Management, Risk Management, Change Management, Verification and Validation, reviews, and Configuration Management and Tracing of Requirements.

In the context of Requirements Engineering, Quality Control may also focus on verifying whether the produced requirements documentation meets relevant quality criteria.

When planning the Quality Assurance activities for a project, it is important to remember that there are some factors which can affect the desired quality level of the product. Such factors include:

- The Customer’s quality criteria or expectations related to the QA process
- The type of product that is being produced (e.g., the complexity or target audience of the product – a product intended for a small audience might have a lower level of some quality attributes than a mass market product)
- The environment in which the product is produced (e.g., the technical environment might limit the ability to achieve the desired level of quality attributes)
- Domain (e.g., complexity of the business domain, the level of innovation, the frequency of changes in the business)
- Legal, safety and environmental factors (e.g., regulations resulting in a higher level of quality to be achieved)

- Time and cost pressures that reduce the ability to execute the Requirements Engineering processes

In order to ensure the required level of quality, verification and validation should be planned and executed from the beginning of the project (refer to Section 5.5 Requirements Validation and Verification).

The following tools and techniques may be used for Quality Assurance and Quality Control of requirements:

- Standards and templates
- Traceability
- Different types of reviews
- Prototyping
- Observance of requirements/specification quality criteria

Quality Assurance can be also supported by ensuring the testability of requirements. Testable (verifiable) requirements can be checked if they are implemented correctly and according to the stakeholders' needs. Testability of requirements is supported by the Acceptance Criteria for a project (refer to Section 5.5 Requirements Validation and Verification).

The quality of a requirements/solution specification may be improved by including the following elements:

- The purpose of the document, scope, definitions and glossary
- The objectives for the different levels (for example, a high-level requirements specification has different objectives than a detailed functional requirements specification)
- Design and implementation constraints
- Priorities or grades for the requirements
- Clear statements of what a system should do rather than how it should do it
- Documentation for any applicable legislation, assumptions, business rules and dependencies
- No supplementary descriptions of diagrams that are clear and can stand alone (replace difficult, abstract text with diagrams where possible)
- Clearly specified user catalogue and permission scheme (users rights and privileges)
- Structured presentation
- Simple, clear, precise and unambiguous language

As stated in Section 3.2 Generic Requirements Engineering Process, requirements are the basic input information to the process of developing and testing the system. Well-defined requirements reduce the risk of project, and particularly product, failure as they support careful testing. Stable requirements increase the chances of meeting deadlines.

Requirements Engineering is closely connected to testing. Good test cases require good, testable requirements. The involvement of testers during the development of the specification is, therefore, very important. The products of Requirements Engineering support testing by providing the basis for testing. Requirements and their specifications are sometimes referred to as the test basis.

It is important to remember that the requirements should be validated through static tests (reviews) which may involve testers. Acceptance by the Test Manager(s) may be required, particularly when the requirements have been found to be untestable. Testers help improve the quality of requirements by identifying weak points and possible defects. Testers should also participate in reviewing requirements to ensure their testability.

Ensuring and controlling quality usually requires metrics. A metric is a measurement scale and the method used for measurement [ISO 14598]. Metrics make it possible to make a quantifiable statement regarding the project status and quality. It is important to remember that results of measurement (numbers collected during the measurement) must always be compared to reference data (a relevant metric).

Sample metrics that can be applied to requirements are the following:

- Project costs
    - Number and complexity of requirements
- Project stability
    - Number of changed requirements
- Number of defects found in requirements or solution specifications
    - Type of defect
    - Number of defects with different types (e.g., logic, consistency, data defects.)

Measurement of the quality of requirements is usually difficult as there is no simple way to express quality. Therefore quality is usually measured by checklists covering common quality attributes. Some questions to be used when evaluating the quality of requirements include the following:

- Is the requirement correct?
- Is the requirement unambiguous?

- Is the requirement feasible?
- Is the requirement traceable?
- Is the requirement identifiable?
- Is the requirement testable?
- Is the requirement design independent?

The change rate can also be a measure for the quality of requirements (this does not apply to Agile projects). Changes are made to clarify unclear requirements and correct incomplete or inconsistent requirements. The higher the change rate of the whole set of requirements, the more a project is at risk. Therefore the change rate should be measured to manage risks within a project.

For training companies: provide examples of metrics that measure the quality of the requirements and the Requirements Engineering process.

5 Requirements Development	290 minutes

Terms

Acceptance Criteria, Apprenticing, BPMN, Brainstorming, Conceptual Model, Contract, Degrees of Formalization, Delphi Method, Field Observation, Function Point Analysis, Goal, GUI Modeling, Identification on the basis of existing documents, Interview, Model, Planning Poker, Prioritization, Questionnaire, Requirements Elicitation, Requirements Model, Requirements Specification, Representative of the Customer, Reuse, Self-recording, S.M.A.R.T., Solution Model, Solution Specification, Specification, SysML, UML, Use Case, Use Case Points Analysis, User Story, Vision, Workshop

Learning Objectives for Foundation Level of Requirements Engineering

The objectives identify what you will be able to demonstrate following the completion of each module.

5.2 Requirements Elicitation

LO-5.2.1	Understand the goals of Requirements Elicitation (K2)
LO-5.2.2	Explain the typical sources of requirements and provide some examples (K2)
LO-5.2.3	Understand the meaning of a typical project vision for Requirements Engineering (K2)
LO-5.2.4	Understand the role of stakeholders for Requirements Elicitation and explain how stakeholders can be identified (K2)
LO-5.2.5	Explain the application of various techniques for Requirements Elicitation and apply them for a given scenario (K3)

5.3 Requirements Analysis

LO-5.3.1 Explain the goal and main activities of Requirements Analysis (K2)

LO-5.3.2 Explain the difference between requirements and solutions (K2)

LO-5.3.3 Recall the reason for effort estimates and provide examples of different estimation approaches (K2)

LO-5.3.4 Apply the procedure for requirements prioritization for a given scenario (K3)

LO-5.3.5 Explain the purpose and consequences of requirements acceptance (K2)

LO-5.3.6 Recall the different models used in Requirements Analysis (K1)

LO-5.3.7 Understand the application and be able to create basic analysis models (context analysis, functional decomposition, business process models) (K3)

LO-5.3.8 Explain the characteristics of basic UML diagrams (state transition diagrams, use case diagrams, activity diagrams, and class diagrams) and their possible application (K2)

5.4 Requirements Specification

LO-5.4.1 Understand and explain the usage of use case and user story in development of requirements specification and solution specification, (K2)

LO-5.4.2 Understand and explain the purpose, characteristics, contents and typical procedure for creating a solution specification (K2)

LO-5.4.3 Describe the common characteristics of different levels of formalization and explain when a specific level of formalization can be used (K2)

5.5 Requirements Validation and Verification

LO-5.5.1 Summarize Verification and Validation techniques for avoiding requirement errors (K2)

LO-5.5.2 Understand the use of Acceptance criteria (K2)

5.1 Introduction to Requirements Development	10 minutes

As defined earlier, the Requirements Development process covers the following activities:

- Requirements Elicitation
- Requirements Analysis
- Requirements Specification
- Requirements Validation and Verification

These activities should not be considered as individual phases without any interconnections. Very often there is a high market pressure for shorter development cycles together with the risk of frequent changes, upgrades or revisions of the solution under development. Therefore, it is nearly always impossible to implement the Requirements Development process as a linear process in which requirements are collected from the stakeholders, analyzed, specified and handed over to the development team which then creates the solution model and validates the requirements. In reality, requirements typically iterate towards a level of quality and detail that is sufficient to permit design and procurement decisions to be made. In almost all cases, the requirements are going to change, which means that proper steps have to be taken to mitigate the effect of the changes.

5.2 Requirements Elicitation	100 minutes

The first step in the Requirements Development process is usually Requirements Elicitation. The main aim of the Requirements Elicitation is to collect requirements from all possible stakeholders – not only users and sponsors, but the project team, the market and other external sources as well. In general, the main purposes of Requirements Elicitation are the following:

- Identifying all desired functions, characteristics, limitations and expectations
- Orientating the requirements toward the project vision
- Detailing high-level requirements and describing functions and services clearly
- Excluding functions and features that the customer does not want

Typical sources of requirements include the following:

- Documents
- Systems in use
- Stakeholders

These sources can influence the chosen technique for Requirements Elicitation.

Requirements Elicitation is not only collecting stakeholders' needs by asking questions – very often the information collected has to be interpreted, analyzed, modeled and validated before a complete set of requirements for a solution can be established. The elicitation techniques and tools to be used are sometimes driven by the choice of the modeling diagrams or the general analysis approach. Many modeling techniques imply the use of a particular kind of elicitation technique as well.

The first step in Requirements Elicitation is to find out what problem is to be solved. This includes identifying stakeholders and establishing (or understanding previously identified) high-level business goals. Stakeholders provide the requirements and limitations; therefore, it is very important to identify all of them. Business goals support maintaining the vision of what is to be done – requirements collected during Requirements Elicitation should support meeting the business goals. Aligning the requirements with the business goals also helps to control the solution scope.

Initial requirements are provided by the customer. The customer is one of the key stakeholders of the project as his needs must be satisfied.

The customer must always be involved. The goal is to develop a mutual understanding with the customer. The supplier should always put himself in the customer's position.

One of the most important sources of requirements is a contract between the supplier and the customer. The agreement (contract) should formally specify and describe what the customer wants. It must be ensured that the interest of the customer takes center stage (which implies that the suppliers should not force a solution they prefer but should analyze the customer's needs and recommend a solution that satisfies those needs in best possible way).

The contract must be in accordance with using the available resources to implement the solution and should be based on: estimations, deadlines, prices and project plans. Requirements Engineering provides input information for such estimates.

The contract should include:

- A short description of the planned solution
- The list of prioritized requirements
- The acceptance criteria for each requirement
- The list of products to be supplied (e.g., documentation, code, working software, services, processes)
- The schedule for the deadlines for development and delivery of the product
- Other needs and expectations such as the preferred technology to be used, resource requirements, etc.

Another important source that expresses the customer's primary needs is the project vision. For each project, the vision must be set anew. A vision should define the objectives to be achieved in high-level form. It is crucial to have a clear definition of the project vision as the vision provides a way to verify if the project provided the required value, or not.

A vision is realized by goals to be met by a specific project. Goals should express the business objectives of the project and should be S.M.A.R.T. to allow measuring if they are achieved after completing the project. S.M.A.R.T. is a method that is used to establish goals and define their quality objectives. S.M.A.R.T. requires that all goals have the following characteristics [G. T, Doran]:

- S – Specific
- M – Measurable
- A – Attainable
- R – Relevant
- T – Timely

When eliciting requirements, it is important to ensure that the requirements will meet both the project vision and the defined S.M.A.R.T. goals.

The following processes can help to establish visions and goals:

1. Finding sources for requirements
2. Analyzing the current situation (objective evaluation)
3. Evaluating the current situation (subjective factors being added)
4. Deducing the visions/goals (based on cost-benefit calculations)

To be able to identify all requirements and needs, all stakeholders on the customer and supplier side must be identified. Each stakeholder or each group of stakeholders may provide new requirements and influence the design of the planned solution. If not all stakeholders are identified, there is a risk that some important requirements or limitations remain unknown and are not considered in the design. Missing stakeholders may result in complex changes being requested for the product in the later stages of the project or after releasing the system into the production environment.

Stakeholders should be described in the following terms: name, function, availability, domain, goals and interests.

Some stakeholders may create interest groups (e.g., all business stakeholders). Interest groups should be brought together as it allows managing their requirements more effectively.

The procedure of identification and evaluation of stakeholders involves the following activities:

- Identification of stakeholders (done by analyzing the business processes, determining process and product owners, analyzing organizational structure and market)
- Grouping stakeholders into groups and selecting representatives of specific groups (if possible)
- Determining relationships between stakeholders
- Identifying potential conflicts
- Analyzing conflicts, determining their sources and identifying win-win opportunities
- Identifying risk-minimizing stakeholders to involve them more in project activities
- Identifying the stakeholders' perspectives

When all stakeholders, business goals and the vision are known, it is possible to start with detailed Requirements Elicitation.

The most common techniques for Requirements Elicitation are the following:

- Questionnaires
- Interviews
- Self-recording
- Solicitation of information from representatives of the customer on site
- Identification on the basis of existing documents
- Reuse (Reusing the specification from another project)
- Brainstorming
- Field observation
- Apprenticing
- Workshops
- Use cases

Technique	Description	Application	Disadvantages
Questionnaires	A questionnaire can consist of open-ended or closed-ended questions. An open-ended question requires the respondent to formulate his own answer. A closed-ended question requires the respondent to choose an answer from a number of possible options. These options should be mutually exclusive.	To confirm or detail previously known requirements; organize content of requirements; select options for requirements/solution	Not applicable for collecting implicit knowledge Low return rate with no responders' motivation Can be directive, which prevents identification of real user needs

Technique	Description	Application	Disadvantages
Interview	A conversational technique where the interviewer asks questions of the responder to obtain information on specified topics. This technique is very interactive and allows modifying the order of previously prepared questions depending on the responder's answers and the situation. Good interviews are more difficult to conduct than one would think due to normal conversation behavior (e.g., ending phrases for the conversation partner), which can lead to interpretations being introduced into the data. The interviewer should ask open questions to obtain information and close questions to confirm or reject specific options.	To obtain information on a specified topic; to clarify requirements.	Time-consuming Insufficient reproduction of results (difficulty getting the same answers when repeating an interview)
Self-recording	In this technique the stakeholder (for example, an end-user) documents the activities he performs to complete a specific task. In addition to documenting the activities "as-is" the user also describes changes, desires and needs. This technique is often associated with demonstrations or document reviews.	To understand complex procedures or processes, especially when the Requirements Engineer is not able to watch the user performing his tasks.	Neglecting "automated" activities Highly dependent on the motivation and the experience of the user

Technique	Description	Application	Disadvantages
Representatives of the customer on site	This approach is one of the most effective Requirements Elicitation (and validation) method as it allows the representative to systematically monitor the progress, verify the correctness of the design and provide feedback and additional information whenever necessary. Having representatives of the customer on site is one of the main rules in Agile methods.	To collect and manage requirements in Agile approaches; to provide user-oriented requirements which can be easily accepted.	High costs for the customer Adaptation costs
Requirements identification on the basis of existing documents	This technique can be used when there is existing documentation that can help in identifying requirements within an organization. Examples of this documentation include process models and maps, process descriptions, organization charts, product specifications, procedures (i.e., work procedures), standards and instructions, templates of documentation, etc. The requirements identified are the basis for further Requirements Analysis and need to be detailed and extended with other, related and linked requirements.	To collect requirements for a solution that will cover existing business processes.	Not applicable when there are only basic documents or no documents within an organization. Not applicable when the documentation is not maintained correctly (not kept up to date).

Technique	Description	Application	Disadvantages
Reuse (Reusing the specification of a certain project)	Reusing the specification of a similar project can be done when an organization already completed one or more projects similar to the current project. A requirements specification prepared for a previous project(s) can be used in another project to shorten the duration of Requirements Analysis and the effort to prepare documentation which should allow starting the implementation earlier. In most cases only some parts of existing specifications can be used for a new project. The documentation to be reused should be always checked for compliance with the current needs and requirements and adjusted accordingly.	For projects aiming to provide products customized for specific customers; to shorten Requirements Engineering in projects developing similar solutions.	High costs of the first project. Requirements that are too detailed, resulting in extensive and costly Change Management.
Field observation	Field observation consists of watching the users' activities and processes, and identifying the system requirements on that basis. On-site observation is conducted by watching the users working and documenting the processes, tasks and results. In some cases, observation is extended to interviewing users about their jobs and how they accomplish their tasks.	To collect accurate requirements by observing users at work; to avoid problems which the stakeholders have with expressing their needs; to improve existing business processes or systems.	Exceptional cases may be omitted Not applicable in some situations (for example, for safety and legal reasons)

Technique	Description	Application	Disadvantages
Apprenticing	The purpose of apprenticing is to collect requirements from a customer, especially when the processes and activities performed by the customer's staff are not easy to describe using other techniques, such as interviews, or the customer has problems with articulating the requirements. Apprenticing is a process of learning about the job from the customer. The customer, who knows best how to do a specific job, teaches the requirements engineer – like a master and a student.	To understand complex business processes to be able to propose the best solution; to help overcome the difficulty that the customer's employees may have in thinking abstractly and expressing their tasks in words.	High cost and time consuming Not applicable in dangerous environments
Use cases	Use cases allow the requirements engineer to capture the functionality of the solution from the perspective of the Actors. An actor can be either an end user or an external system. Use case diagrams define the boundaries of the solution and show the connections with external systems and actors.	To collect and explain requirements from the user's point of view; to clarify and organize the functionality that is requested from the solution.	Not applicable in case of solutions consisting mostly of non-functional requirements

Technique	Description	Application	Disadvantages
Workshops	A workshop is a kind of meeting that is focused on a specific (previously defined and announced to the participants) topic, usually involving stakeholders representing different areas and/or domains for a short, intensive period. Workshops involve people who have different points of view on a given problem and help to determine and describe requirements coming from various perspectives. Workshops are one of the key practices in Agile methods as they involve all the main stakeholders in developing the Product Backlog. Properly prepared and executed workshops help to mitigate possible difficulties such as remote connections to manage geographic distribution, subgroup sessions to mitigate lack of consensus and so on.	To identify requirements in order to establish a solution's scope; to discover hidden requirements (i.e., requirements that are not directly stated or even realized by the stakeholders but are needed to accomplish some of their needs or higher level requirements); to develop or detail requirements in a newly identified area; to define priorities of requirements or reach consensus when it comes to reaching agreement on the requirements (sign off); to discover and resolve potential conflicts between stakeholders' requirements; to review results of specified processes or activities and resolve issues that may have appeared.	Difficult in case of geographically distributed teams Availability of all people required to attend the workshop Consensus is not necessarily easily reached during a workshop, and discussion can stall on (minor) problematic issues, thus making the process lengthy and de-motivating for participants.

Technique	Description	Application	Disadvantages
Brainstorming	Brainstorming is a commonly used technique to obtain requirements related to areas of an organization's activity or planned system functionality that are new or not well known. It allows collecting many ideas from various stakeholders in a short time and with low cost. During the brainstorming session the participant are submit ideas and concepts regarding a given problem.	To resolve requirements conflicts; to define various options of a solution; to resolve business problems (for example, low efficiency of a process).	Difficult with non-motivated participants Difficult to apply in distributed teams

Table 3 Requirements Elicitation techniques

For training companies: provide examples of applications of at least three different techniques for Requirements Elicitation.

To achieve effective results and avoid requirements gaps, usually a combination of the elicitation techniques described above is used.

As stated earlier, functional requirements specify the functions of the system as they are perceived by the end user. Functional requirements also describe triggers of the process such as user actions or input/output data that cause the business process to start.

When eliciting requirements, it is important to ask not only about functions, but about quality attributes as well. Non-functional requirements (NFR) describe the quality attributes of the whole system or its specific components or function. They can limit the solution, for example, by requiring specific efficiency parameters. Non-functional requirements are difficult to describe and are often vaguely expressed or not documented at all making them difficult to test. Particular attention should thus be paid to non-functional requirements at all stages of the RE process to be sure they are expressed clearly and are measurable.

Non-functional requirements can describe different aspects of the solution performance. According to ISO/IEC 25000 (previously ISO 9126), specific types of non-functional requirements are the following:

- Reliability

- Usability
- Efficiency
- Maintainability
- Portability

Non-functional requirements are especially important as they specify criteria that can be used to judge the operation of a system; therefore, they have a great impact on the customer's satisfaction in using the product. Functional requirements have to provide functions; non-functional requirements determine how easily and effectively the functions can be used.

For training companies: provide examples of functional and non-functional requirements according to ISO/IEC 25000.

Requirements, once elicited, should be properly documented to allow further tracking and Requirements Analysis.

Requirements must be clearly and accurately specified. They should be measurable in order to ensure they are testable and their implementation can be properly checked. It is important to remember that common language has some limitations and disadvantages. This may cause the description of the requirements to be unclear and ambiguous. Therefore, proper standards and templates should be used wherever possible. Standards provide a common understanding and the best practices for writing the specification when templates limit the language that can be used.

In addition to standards and templates, vocabularies are an important tool to facilitate communication between different stakeholders and to introduce some control over the natural language's ambiguity.

The description of a requirement must satisfy quality criteria (refer to 1.1.4 Quality of requirements).

Depending on the abstraction level, requirements can be described with less or more detailed. In some development models, business requirements can be written in the form of high-level use cases (for example, Rational Unified Process), or user stories (Agile approaches). In general, the typical structure of a business requirement statement should cover the following aspects:

- The user – who would like this requirement?
- The result – what is the result the stakeholders are looking for?
- The object – what is the object the requirement addresses?
- The qualifier – what is the qualifier that is measurable?

For example, the following requirements statement: “The insurance agent must have information about any new products one day prior to the product launch.” covers the four elements listed above in the following way:

- The insurance agent ← who
- must have information about ← what result
- any new products ← what object
- one day prior to product launch ← qualifier

More detailed requirements (solution/system requirements) can be described in a form of system use cases, often together with scenarios or as user stories in Agile approaches. In general, solution/system requirements should cover the following aspects:

- Affected process – what is the business process the requirement is referring to? This element usually focuses on functionality and determining inputs and outputs.
- Solution activity – how is the activity triggered? Is it an independent system activity, or an interaction with the user, or an interface with other systems/processes?
- Legal commitment – what is the legal commitment of the requirement? Common key words are often used to describe it (should, must, etc.).
- Logical and time constraints – are there any boundary conditions applicable for the requirement?

For training companies: discuss examples of descriptions of business requirements and solution/system requirements.

Identified and analyzed requirements are usually documented in the form of a specification (refer to 5.4 Specification of Requirements).

5.3 Requirements Analysis	80 minutes

The next step after Requirements Elicitation is Requirements Analysis. The main goal of analysis is to create a business solution that will implement the requirements. Creating the business solution requires elaborating the customer needs into system/solution requirements and down to product-component requirements. It also includes detecting and resolving conflicts between requirements both at the same level and between different levels, and discovering the boundaries for the solution including how it must interact with its environment.

The high level and iterative procedure for Requirements Analysis contains the following steps:

1. Analysis of the needs
2. Description of the solution
3. Effort estimate and prioritization
4. Requirements acceptance

The first step aims to analyze the stakeholders' needs to be able to propose a solution meeting those needs. At this time Requirements Analysis often includes verification and validation activities to ensure that the requirements are correctly understood and approved by the stakeholders. An important part of the Requirements Analysis is checking the clarity and consistency of the requirements and resolving any conflicts.

The description of the solution is usually done with models. In general, there are three basic types of analysis models: requirements models, solution models, and conceptual models. Models are developed through suitable methods of analysis.

Another step in Requirements Analysis is the effort estimation and prioritization of the requirements.

Effort estimates connect Requirements Engineering with Project Management. Early effort estimates are based on high-level requirements and are used to estimate the cost of the project. During the solution development, when requirements evolve, the initial cost estimates are used to control the project and – in case of changes – to estimate the effort and cost of the changes.

Estimations are usually focused on the following:

- Costs
- Time
- Requirements

The cost of the project depends on various factors, including:

- Project type
- Process maturity
- Design and testing approaches, methods and tools
- Technology used
- Complexity of the planned solution
- Quality objectives (for example, the desired level of product quality)
- Team qualifications
- Team distribution
- Experiences

The accuracy of the effort estimation depends upon the progress of the project and its maturity.

In general, effort estimation can be conducted using analogical conclusion and algorithmic procedures. Both approaches require that estimation procedures are based on historical data and framework conditions.

Analogical conclusion estimates effort based on comparison with similar projects. The estimate is based on experience, not on mathematical formulas. With this technique the current project is compared with past projects. The comparison may include:

- The number and complexity of requirements
- Business stability
- Scope of the solution
- Technology used
- Characteristics of the personnel (e.g., skills, experience)

The following are sample analogical procedures:

- Planning Poker
- Delphi method

An example of an analogical procedure can be found in Agile projects. In such projects, especially when managed by Scrum, there is an estimation method called Planning Game or Planning Poker. This method is used to gain consensus among the team. The team ability is then measured through something called the Burn Down Rate. Retrospective sessions are conducted after every

sprint where the estimated points are compared to the actual time required. This improves the team's ability to estimate.

The Delphi method is another example of an analogy technique. It is a structured communication technique used to conduct interactive forecasting. It involves a panel of experts [Linstone75]. The experts are asked to answer questionnaires in some number of rounds. After each round, there is an anonymous summary of the forecasts made by the experts together with the reasons for their judgments. The experts then revise their earlier estimates by taking into account replies from other members of their panel.

It is believed that during this process the range of the answers will decrease and the group will converge toward the "correct" answer. The process is stopped at a pre-defined stop criterion. The mean scores of the final rounds determine the results.

The second approach to effort estimation is an algorithmic procedure. In this case the efforts are calculated on the basis of parameters. Parameters can describe the product (volume, duration), the boundary conditions (efficiency), etc. Sample algorithmic procedures include the following:

- Function Points Analysis
- Use Case Points Analysis

Function Points Analysis requires counting so called function points in the planned solution. A function point is a unit of measurement to express the amount of business functionality provided by an information system to a user. The cost of a single Function Point is calculated on the basis of past projects. Identified functions are weighted according to three complexity levels; the number of points assigned to each level differs depending on the type of the Function Point.

Use Case Points Analysis is quite similar to Function Points Analysis but it requires analyzing among other items the number and complexity of use cases and actors in the planned solution.

Another important activity of Requirements Analysis is prioritization. Prioritizing establishes a requirement's relative importance for the implementation. This allows the team to complete the most crucial requirements first. Prioritization supports incremental development as requirements can be grouped by priority and the highest priority requirements can be scheduled for implementation in the early iterations.

The procedure of establishing requirement priorities involves the following activities:

1. Requirements grouping – which includes identifying requirements that influence or are dependent upon each other (e.g., requirements that create one complex set of functionality).

2. Requirements analysis – analyzing the grouped requirements by all involved stakeholders in order to agree on the level of importance and establish the priorities of the requirements. Impact analysis is often used to support the analysis.
3. Creation of the requirements project plan – establishing a plan where requirements with high priorities are to be developed first and assigning responsibilities for the implementation.
4. Planning increments of system testing - designing test cases for testing each solution increment based on the priorities of the requirements for that increment.

A common approach to prioritization is to group requirements into priority categories. Often a three-level scale is used (i.e., High, Medium and Low). As such scales are subjective and imprecise, all involved stakeholders must agree on the meaning of each level in the scale they use. The definition of the priority should be clearly stated and should be a key attribute of each requirement.

For training companies: explain the procedure of requirements prioritization with some examples. Explain the meaning of requirements prioritization for release/iteration planning.

Requirements acceptance (also called agreeing on requirements or signoff) is the next element of Requirements Analysis. It is a formal agreement that the content and scope of the requirements are accurate and complete. It is very important to ensure that the requirements have been accepted at different phases of the solution development as formal agreement is the basis for further activities. Agreement on the high-level requirements (business requirements) should be obtained before the project starts. Agreement on the detailed requirements (solution/system requirements and component/function requirements) should be obtained, with signoff, before moving to the implementation phase. Obtaining requirements signoff is typically the final task of the Requirements Analysis and Specification phase of the project.

The requirements signoff usually involves the key project stakeholders, including:

- Project Managers
- Customer's business representative
- Business and System Analysts
- Requirements Engineers
- Representatives of Quality Assurance, testing and development teams

One of the purposes of requirements acceptance is ensuring the requirements are stable and that any changes are managed via formal Change Requests. Formal agreement reduces the risk of introducing new requirements during or subsequent to implementation.

Formal acceptance can also minimize the risk of misunderstandings between the customer and the supplier regarding the project's scope as all requirements or requirements/solution specifications should have been carefully reviewed before signoff.

Requirements agreement is considered to be complete, when all the relevant project stakeholders have signed off the requirements document.

Completion of requirements acceptance should be communicated to the project team and usually is a project milestone.

5.3.1 Solution Modeling

Solution Modeling is often a part of Requirements Analysis and Specification as it aims to develop models of a real-world problem to be solved by the specific business solution. Solution Modeling also can be performed during Requirements Elicitation as some of elicitation techniques use different modeling notations or approaches (e.g., Requirements Elicitation with UML use cases). The main challenge for the Solution Modeling is to choose and develop appropriate diagrams which can form a model describing the business solution from different points of view. These models should not only correctly represent the business solution, but should be understandable by both the customer and the supplier representatives.

Solution Modeling can use several types of models, but in general three basic levels of models exist:

- Requirements model – which describes the problem area and is usually designed at the early stages of the project. This model mainly serves for Requirements Analysis and effort estimation and provides a basis for the solution model.
- Solution model – describes the solution area from different points of views and determines the shape of implementation of the functional and non-functional requirements. The business solution model provides a basis for the solution design.
- Conceptual model – also known as a domain model. A conceptual model represents concepts (entities) and relationships between them. The aim of conceptual modeling is clarifying and expressing the meaning of terms and concepts used by domain experts to address the business problem, and establishing the correct relationships between the different concepts.

Different models may be used for the above levels depending on the point of view to be presented via the specific model. Common perspectives applicable to modeling the problem or solution domain include the following:

- User view (e.g., modeled from use cases)
- Logical view (e.g., modeled from the functional requirements)
- Process view (e.g., modeled from communication, interaction models or non-functional requirements specifying the effectiveness of the business processes)
- Implementation view (e.g., modeled usually from the components of the solution)
- Installation view (e.g., modeled from integration models and solution architecture)

Different levels of modeling and different views of the solution can be described by different diagrams. To have the full image of the solution, usually a combination of different views is used. This results in using different diagrams describing the solution model from specific perspectives. Some sample analysis models are presented in the table below (Table 4 Sample analysis methods and models).

Analysis Method	**Analysis Model**
Context analysis	Context model Deployment model (UML)
Logical analysis	Functional decomposition Requirements diagram (SysML)
Decision analysis	State machine model (UML)
Data analysis	Entity-relationship model (ERM) Class model (UML)
Use case analysis	Use case model (UML)
Process flow analysis	Business process model (BPMN) Activity model (UML) Communication model (UML)

Table 4 Sample analysis methods and models

For training companies: explain the benefits and application of basis analysis models (context models, functional decomposition, business process models) and provide some examples of using these diagrams.

One of common methods of software solutions modeling is UML (Unified Modeling Language). UML provides several types of diagrams to describe different views of the solution. UML is a unified notation for the analysis and design of systems. UML provides different diagrams divided into behavior and structure diagrams, where behavior diagrams depict behavioral features of a system or business process, and structure diagrams depict the structural elements composing a system or function.

Diagram Type (UML 2.4.1)	**Application**
Behavior Diagrams	
Activity Diagram	Model the behaviors of a system, and the way in which these behaviors are related in an overall flow of the system. Usually used to model procedures, process flow, and workflows.
Use Case Diagrams	Capture Use Cases and relationships between Actors and the system. Describe the functional requirements of the system, the manner in which external operators interact at the system boundaries, and the response of the system.
State Machine Diagrams	Show how an element can move between states, classifying its behavior according to transition triggers and constraining guards.
Timing Diagrams	Define the behavior of different objects within a time scale. Provide a visual representation of objects changing state and interacting over time.
Sequence Diagrams	Structured representation of behavior as a series of sequential steps over time. Used to depict work flow, message passing and how elements in general cooperate over time to achieve a result.

Diagram Type (UML 2.4.1)	Application
Behavior Diagrams	
Communication Diagrams	Show the interactions between elements at run-time, visualizing inter-object relationships.
Interaction Overview Diagrams	Visualize the cooperation between other interaction diagrams to illustrate a control flow serving an encompassing purpose.
Structure Diagrams	
Class Diagrams	Capture the logical structure of the system, the classes and objects creating the model, describing what exists and what attributes and behavior it has. Can be used to model data.
Object Diagrams	Present instantiations of classes and their relationships at a point in time.
Package Diagrams	Depict the organization of model elements into packages and the dependencies between them. Can be used to organize requirements according to their types/levels of abstraction.
Component Diagrams	Present the pieces of software creating a system as well as their organization and dependencies.
Deployment Diagrams	Describe the execution architecture of the system.
Composite Structure Diagrams	Reflect the internal collaboration of classes, interfaces and components (and their properties) to describe a piece of functionality.
Profile Diagrams	Allow to tailor the UML metamodel for different platforms (such as J2EE or .NET) or domains (such as real-time or business process modeling).

Table 5 UML diagrams and their application

To model more complex solutions, especially in System Engineering, another unified modeling notation can be used – SysML (System Modeling Language). SysML allows modeling a wide range of systems which include hardware, software, information, processes, personnel and facilities.

SysML reuses seven of UML's diagrams and provides two new diagrams: a requirement diagram which captures functional, performance and interface requirements and a parametric diagram to define performance and quantitative constraints.

To model business processes flows and communication between business actors, other modeling notations can be used – for example BPMN (Business Process Modeling Notation) which is usually used in early phases of Requirements Analysis or during analysis of business processes performed within the customer's organization.

To model user interface aspects of the planned solution (especially in the case of software solutions), GUI modeling can be used. It is usually done via so called GUI prototyping and provides a way to design and test elements of the user interface. GUI modeling is especially useful in cases where the requirements are not clear or can be implemented in many different ways, as it allows the stakeholders to choose the best option of implementation.

The following different types of prototypes can be distinguished [IBUQ]:

- Vertical prototypes: reduction to a few individual, but detailed functions.
- Horizontal prototypes: as many functions integrated as possible, but not functional (mostly used for user interface testing).
- Scenario prototypes: all functions are simulated for a specific task using a combination of vertical and horizontal prototypes.

Depending of the purpose of use, prototypes are created in different forms and variations [IBUQ]:

- Low fidelity (lo-fi) prototypes – low similarity to the end product; review the idea's benefit
- High fidelity (hi-fi) prototypes – high similarity; review of details and exact functions
- Hybrid forms – the low-high fidelity (lo-hi-fi) prototypes, for example, interactive simulations using HTML or PowerPoint®

For training companies: explain the benefits and application of basis UML diagrams (activity diagrams, use case diagrams, state machine diagrams and class diagrams) and provide some examples of using these diagrams.

5.4 Requirements Specification	60 minutes

Requirements Development is not only a process of discovering and analyzing requirements, it is also a process of facilitating effective communication of these requirements among different stakeholders. The way in which requirements are documented plays an important role in ensuring that they can be read, analyzed, changed and validated. Documenting requirements is called Requirements Specification.

A specification is defined as an explicit set of requirements to be satisfied by a material, product, or service; therefore it can be considered as a kind of contract describing the solution behavior and features. The specification serves to track and manage requirements. In the specification, requirements are specified in a structured way and are modeled separately (requirements are modeled “independently”, as high-level requirements are broken down to a level in which each individual requirement constitutes an "independent" entity that can be further developed and tracked). The specification is a document on which a formal agreement on the requirements to be implemented in the planned system (or in other forms of the solution) is based, therefore Requirements Specification can be considered as a part of or the final result of Requirements Analysis.

The term “specification” may be used in the context of requirements and solutions.

Requirements Specification describes the problem area of interest (a business solution proposal for a given business problem, need or objective, a new feature, etc.) and contains at least the following information:

- Business requirements together with their acceptance criteria
- Limitations and assumptions

The creation of the customer’s requirements specification should be the customer's task. However in some cases, the supplier can support preparing the requirements specification.

A requirements specification does not have to be a formal “specification” document. For example, it could be a sprint backlog or a set of requirements maintained in a requirements management tool.

Creation of other types of specifications for requirement such as system requirements, software requirements, safety requirements, security requirements, environmental requirements, legal requirements, etc. are tasks for other roles.

The solution specification describes the solution from different points of view. The specific types of solution specifications include: functional requirements specifications, system requirements specifications or software requirements specifications.

For example, a functional specification is a document that clearly describes the technical requirements for solution. The functional specification is the basis for further product development and testing. It must provide precise information about all functional aspects of the software to be implemented. A functional specification does not describe how to implement the desired functions of the software product and it does not prescribe what technology to be used. Instead, it focuses on functionality and interactions between the user and the product.

Both requirements and solution specifications can be written in the form of use cases (business use cases or system use cases, depending on the purpose and target audience of the specification).

Another type of a specification is a User Story. User Stories are often used with Agile development methodologies. User Stories are a quick way of handling customer/user requirements. The intention of the User Story is to be able to respond faster and with less overhead to rapidly changing real-world requirements.

A User Story describes the functionality that will be valuable to the customer. It is composed of three aspects [Cohn]:

- A written description of the story used for planning and as a reminder (usually in a form of a statement “As a [end user role], I want [the desire] so that [the rationale]”)
- Conversations about the story that serve to flesh out the details of the story
- Tests that convey and document details and are used to determine when a story is complete

User Stories are often used together with Personas (i.e., archetype characters) representing a specific type of end user role.

Specification work can be supported by standards that provide guidelines on how to write a specific document, its typical contents, or best practices. Common standards that can be used for specification creation are the following:

- IEEE 830. Recommended Practice for Software Requirements Specifications (also known as a standard for SRS).
- IEEE 1233. Guide for Developing System Requirements Specifications (also known as a standard for SyRS).
- IEEE 1362. Guide for Information Technology – System Definition (also known as Concept of Operations, ConOps).

In general, a specification serves as an activity for formalizing the results of the Requirements Analysis. The identification, analysis and specification activities lead to requirements acceptance (refer to 3.3.2 Requirements Analysis).

The procedure for creating a solution specification usually includes the following activities:

1. Definition of the vision and objective

- What is the objective of the solution or the specific area of the solution that is documented in the solution specification?

2. Identification of stakeholders

- Who will use the product?
- Who is responsible for the specific areas of the solution and can support or impact the solution specification creation?

3. Requirements determination

- What high-level (business) requirements should be covered in the specification?
- What are the priorities of the requirements?

4. Structured requirements specification

- What dependencies among requirements are there?
- How can requirements be organized?

5. Description of the system environment

- What is the context of the solution?

6. Determination of the solution

- The solution and scope definition with the relevant aspects outside of, but influencing, the scope such as interfaces to external systems

7. Requirements analysis

- Detailed requirements analysis (decomposition of business requirements into lower level solution/system or function/component requirements)
- Determining and resolving potential conflicts between requirements
- Mapping business requirements to solution/system requirements and creating a structure for traceability

8. Modeling of the business problem

- Modeling of the business problem to be solved by the solution (including modeling or improving existing business processes, if applicable). Business process modeling notations are commonly used to express the business model (for example, BPMN)
- How will the solution be built into the existing business infrastructure?

9. Modeling of the solution

- Modeling of the solution (that may include characteristics of the software, hardware, services, processes, etc. depending on the type of the solution specification)
- Modeling notation to present the solution from different points of view (for example, UML)

The above procedure involves a number of stakeholders who are supporting the specification work in different area and formalizes the results of the Requirements Analysis process. Individual steps performed within the procedure might be done in parallel, iteratively or even skipped, depending on the information already collected and the selected Requirements Engineering approach (for example, in Agile methodologies, the procedure for solution specification is not as formal since requirements and solutions are usually described in light weight way). The output of the procedure, the solution specification, serves as a starting point for software, hardware and database design. It describes the function (functional and non-functional specifications) of the system, performance of the system and the operational and user interface constraints.

Product documentation, including requirements or solution specifications, may be created with different degrees of formalization:

- Non-formal
- Semi-formal
- Formal

A non-formal approach to writing a specification means the document is written in common language, without any formal notation. This approach may be used when the readers have no experience with more formal or technical specification languages and would have difficulties in understanding the content of the document. The main weakness of this approach is that it is ambiguous and may lead to misunderstanding and over interpretation. In addition, non-formal specification is not a good basis for implementation and testing as is not clear and precise enough.

A semi-formal specification will include some formal notation and will be well structured. Usually such specifications are based on specific templates (often derived from relevant standards). Semi-formal specifications may express requirements in the form of models and use formalized common language. One of most common notations used for semi-formal documentation is UML.

A formal specification is a mathematical and/or algorithmic description of a solution that may be used to develop an implementation. A formal specification describes what the product should do and usually does not describe how it should be done. As it is based on mathematical formulas or a specific language (e.g., VDM Specification Language (VDM-SL)) and is more difficult to learn, formal specification is used quite rarely and requires specific knowledge.

5.5 Requirements Validation and Verification	40 minutes

As the requirements are the basis for system development, any error or missing requirement will propagate to the other development processes in the project. It is important to notice that defects resulting from low quality requirements are more expensive to fix in later phases of the project than other types of defects. In addition, the later defects are detected, the higher is the cost to fix them.

Therefore the use of verification ("are we producing the product correctly") and validation ("are we producing the right product") of the requirements are necessary activities. Requirements Validation and Verification should be done continuously during the development of the solution to ensure that the product being developed meets the quality criteria and will satisfy the stakeholders' needs. The best practice is to plan and perform Validation and Verification of requirements from the early phases of solution development – ideally from the Requirements Elicitation. Common techniques for Validation and Verification include different types of reviews, prototyping or presentations of the proposed solutions or requirements to the stakeholders with the goal to receive feedback.

Validation and Verification activities should also include ensuring that the requirements and/or requirements/solution specifications conform to company standards (templates) and are documented and then tested against the quality criteria (refer to 1.1.4 Quality of Requirements). It is also important to validate the models developed during the Requirements Analysis and Specification activities.

As requirements are the basis for solution development and testing, their quality is crucial for the success of the project. Clear, complete, consistent and testable requirements reduce the risk of product (or even more, project) failure, as they allow careful testing. It is therefore recommended to involve testers in reviews of requirements as they can significantly help improve the quality of the requirements and requirements and/or solution specifications by identifying weak points and possible defects.

Testability of requirements is supported by Acceptance Criteria**.** Acceptance Criteria describe criteria that must be met to approve the solution and should be agreed upon by both sides – the supplier and the customer – before starting the project. The best practice is to define Acceptance Criteria as a part of contract documentation. Every high level requirement must have at least one acceptance criterion and each of the criterion must be measurable by a realistic and agreed upon means. Such criteria often create the basis for the project Quality Plan and Acceptance Testing.

For training companies: provide examples of Acceptance Criteria for sample requirements.

6 Requirements Engineering in Models	100 minutes

Terms

CMMI, Agile Approach, Iterative Approach, Maturity Model, Process Model, Product Backlog, Product Life Cycle, Product Owner, Rational Unified Process, Sequential Approach, SPICE

Learning Objectives for Foundation Level of Requirements Engineering

The objectives identify what you will be able to demonstrate following the completion of each module.

6.1 Development and Maintenance Models and Approaches

LO-6.1.1 Summarize the common characteristics of a product life cycle model (K2)

LO-6.1.2 Recognize the similarities and differences in the Requirements Engineering process between common models describing product development or the maintenance process (K2)

6.2 Maturity Models

LO-6.2.1 Understand how Requirements Engineering is a part of common maturity models (K2)

6.1 Development and Maintenance Models and Approaches	60 minutes

The structure of the Requirements Engineering process often depends on the development model used. The generic Requirements Engineering process should be adapted to match the specific needs and the development environment.

The purpose of this chapter is to provide examples of how different Requirements Engineering activities occur in some standard types of development models.

Process models are method-independent descriptions of development processes. Process models usually describe roles, activities, phases and documents and give a standard format for planning, organizing and running a project.

A general product life cycle (PLC) defines various phases of product development where the basic phases are planning, development, maintenance and the end of life. The development phase is often divided into several phases, often performed iteratively (see Figure 2 Basic Product Life Cycle (PLC)).

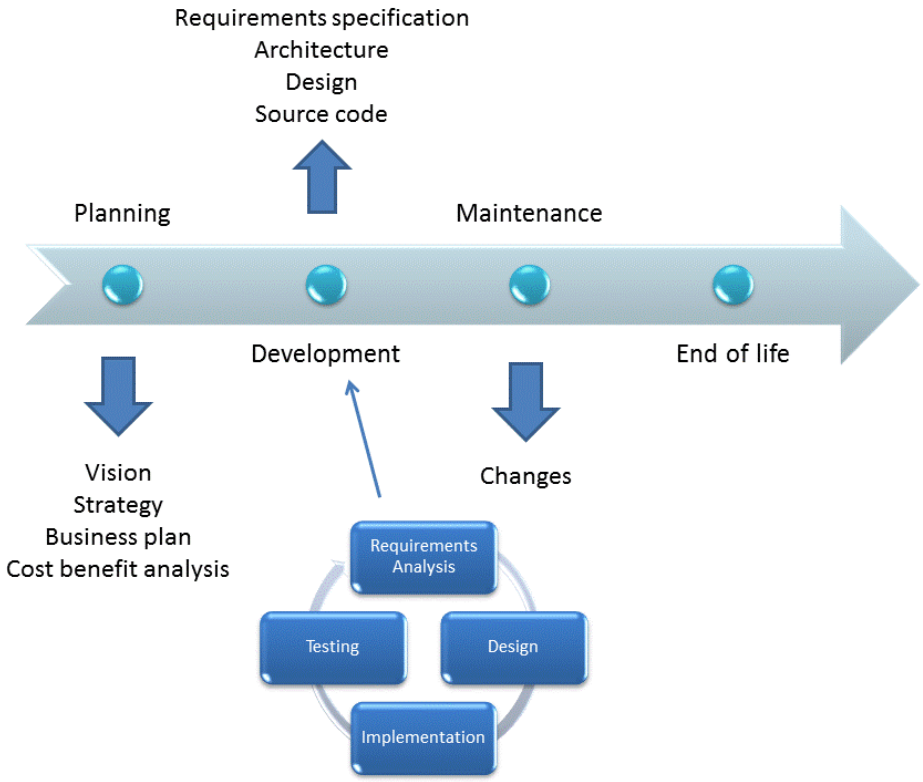

Figure 2 Basic Product Life Cycle (PLC)

This generic PLC is a basis for different development models.

In general, development models can be classified as traditional approaches or Agile approaches. The traditional approaches can be divided into sequential models, iterative models and incremental models (the last two are quite often combined together).

The simplest development model – the Waterfall model – can be regarded as a linear representation of the generic Requirements Engineering process, based on the assumption that it is possible to collect, define and analyze all requirements in an early stage of development and they will not be subject to change. The main weakness of this process model is poor adaptability to changes. This weakness eliminated it from many modern projects where requirements are changing due to business, technology or market changes.

The general V-model has been a further development of the original Waterfall model. The main improvement implemented in the V-model is the distinction between development steps and testing steps, where each development step has a corresponding testing step. The model can be seen as a decomposition of requirements where each level has an associated testing level.

This model is quite effective for Requirements Engineering in the development phases and gives a good basis for Requirements Management with tracking and Change Management. However,

Requirements Development in the V-model can be difficult as it is often impossible to assure that all important requirements are found in the early stages of the project. Very often the initial scope of the product increases due to requirements coming from different stakeholders and surrounding systems or environments in later phases. This causes the need for extensive rework and Change Management.

The V-model can be considered as one of the best models for both Requirements Development and Requirements Management for well-defined and stable areas.

For training companies: graphic portrayal of the General V model; description of Requirements Engineering process in the General V Model

In case of more complex or unstable solutions, it is recommended to use iterative or incremental models. The iterative/incremental approach is built on the fact that requirements are often difficult to specify, especially in commercial systems. Techniques for iterative and incremental development assume cooperation and partnership between developers and users. Requirements are often defined in terms of business objectives only – these business objectives are then elaborated in later iterations.

The incremental development assumes that the list of requirements is divided into smaller parts and each part is developed in steps or increments. The idea is that each part can be implemented and, in the best case, go live so that the time to market is short. It is also possible to get immediate feedback on the delivered increments, so the outstanding, not yet developed increments, can be improved. In this way the risk can be reduced (especially compared to a “one go” strategy). Even if the project is terminated before completion, stakeholders should get some benefits from the already completed increments.

The iterative approach is especially useful for large projects with a huge number of requirements. The main idea of the iterative approach is to develop a solution through repeated cycles (which can be regarded as a mini-PLC) and in smaller portions at a time. The purpose of such organization is to make the project more controllable and to be able to systematically monitor and address risk. An example of an iterative model is RUP©, Rational Unified Process, a process model by IBM Rational.

RUP is a model for software product development in an object-oriented context. It organizes the project into four main phases: inception, elaboration, construction, and transition. Each of these phases has own purposes and products, defined by the RUP framework. RUP is an iterative model where the iteration contents are selected based on the risks of the items being implemented. The number of iterations in a specific phase can differ and is adjusted to the specific needs of the project or product. Requirements Engineering work is generally covered by the Requirements and Analysis and Design disciplines defined by the RUP framework.

For training companies: deepening of the RUP© with graphic presentation; deepened study of the requirements discipline.

All the process models described above belong to so called traditional approaches. The most common challenge for traditional approaches is supporting frequent changes. Most traditional models do not support changing requirements in an effective way. As a solution for this problem, so called Agile models appeared. Agile approaches are based on the Agile Manifesto which presents a set of value propositions and principles used in Agile environments. The most important principles of Agile are [Agile Manifesto]:

- Customer satisfaction by rapid delivery of useful software
- Welcome changing requirements, even late in development
- Working software is delivered frequently (weeks rather than months)
- Simplicity - the art of maximizing the amount of work not done - is essential
- Regular adaptation to changing circumstances

The implications of these principles for Requirements Engineering are quite significant. In Agile environments, requirements are often communicated and tracked through mechanisms such as Product Backlogs, User Stories, story cards, or screen mock-ups. Commitments to requirements are either made collectively by the team or by an empowered team leader. Traceability and consistency across requirements and work products might be addressed through the mechanisms already mentioned (Product Backlogs, User Stories, story cards, and screen mock-ups) as well as during start-of-iteration or end-of-iteration activities such as “retrospectives” and “demo days”. The Requirements Management process is therefore not as formal as it is in traditional development/maintenance models.

The process for Requirements Development is closely related to definition of the specific description and analysis of the User Stories or scenarios. Many of the activities normally performed under Requirements Development are not described in Agile models but are expected to be performed by the Agile Development team under the responsibility of a Product Owner.

In Agile environments, customer needs and ideas are iteratively elicited, elaborated, analyzed, and validated. Requirements can be gathered in backlogs and specified in the form of User Stories, scenarios, or use cases. Which requirements will be addressed in a given iteration is driven by an assessment of risk and by the priorities associated with what is left in the Product Backlog. Which details of requirements (and other artifacts) to document is driven by the need for coordination (among team members, teams, and later iterations) and the risk of losing what was learned. When the customer is on the team, there can still be a need for separate customer and product documentation to allow multiple solutions to be explored. As the solution emerges, responsibilities for the derived requirements are allocated to the appropriate teams.

Sample Agile approaches are Extreme Programming and Scrum.

Extreme Programming is a software development methodology developed by Kent Beck et al with the aim to respond to changing customer requirements. It advocates frequent software releases to the customer in short development cycles (time boxing), which aims to improve productivity and provide checkpoints where new requirements can be adopted.

Some characteristics of Extreme Programming include:

- Avoiding programming of features until they are actually needed
- Expecting changes in the customer's requirements as time passes and the problem is better understood
- Frequent communication with the customer and among programmers
- Complete renouncement of requirement determination (no separate requirements "phase" before development starts; requirements development, refinement and discovery are a part of actual software development (programming))

Scrum is an Agile framework containing sets of practices and predefined roles. One of the major characteristics of Scrum is dividing the development into "Sprints" (typically a two to four week period). During each Sprint, the Team creates a so called "potentially shippable product increment".

Scrum projects manage requirements via "backlogs". There are two types of backlogs:

- Product Backlog – a high-level list maintained during the entire project. It aggregates requirements in the form of broad descriptions of all potential features, prioritized by business value. The Product Backlog is the property of the Product Owner.
- Sprint Backlog – the list of work to be addressed by the team during the next sprint. Features are broken down into tasks which, as a best practice, should normally be between four and sixteen hours of work. The Sprint Backlog is the property of the Team.

Scrum's principal impact on Requirements Engineering is that requirement specifications are not completed and validated before the beginning of development.

The Product Owner and Team agree on features from the Product Backlog to be included in the upcoming sprint based on business priority and required work effort. User requirements are formulated by the Product Owner as "User Stories" that contain information on the "who, what, why" and not the "how" of a requirement. At the beginning of the Sprint, the selected features are broken down into the tasks of the Sprint Backlog, and then developed.

Involvement of the Product Owners, for example viewing presentations of implemented functions of the software, helps clarify requirements for the Team and the Product Owners themselves.

For training companies give examples for user stories, and the corresponding product and sprint back log items.

6.2 Maturity Models	40 minutes

The Requirements Engineering process can be improved just as any other process defined as a part of product/solution development or maintenance. Improvements can be applied to each of the Requirements Engineering activities. For example, Requirements Elicitation may be improved to collect customer requirements in a more effective way. In order to do so, the organization may introduce some techniques that will ensure the requirements are gathered faster, are complete and are agreed upon by most stakeholders. These techniques include interviews, brainstorming, initial prototyping, using Personas, and scenarios.

A common method for improving the Requirements Engineering process improvement is to employ maturity models. These models require assessing the current level of process maturity and identifying areas for improvement. Maturity models often use maturity levels. Maturity levels are used for the identification, assessment and improvement of the process maturity.

Examples of maturity models that can be applied to the Requirements Engineering process are ISO/IEC 15504 (SPICE – Software Process Improvement and Capability Determination) or Capability Maturity Model Integrated (CMMI). Both models define five maturity levels for specific processes or areas and allow comparing the maturity across different organizations.

ISO/IEC 15504 (SPICE – Software Process Improvement and Capability Determination), is a set of technical standards for the software development process and related business management functions. SPICE can be used as a means to process improvement and/or capability determination (for example, evaluation of the supplier's process capability). SPICE defines five maturity levels for the following processes: Customer-Supplier, Engineering, Supporting, Management and Organization. These five levels are:

1 Performed process

2 Managed process

3 Established process

4 Predictable process

5 Optimizing process

The Key Process Areas related to Requirements Engineering in the SPICE model are:

- ACQ1 Acquisition preparation BP1-3
- ENG1 Requirements elicitation BP1-6

- ENG2 System Requirements Analysis BP1-6
- ENG4 Software Requirements Analysis BP1-6

The Capability Maturity Model Integrated (CMMI) defines five maturity levels for development, services and acquisition:

1 Initial

2 Managed

3 Defined

4 Quantitatively managed

5 Optimizing

The Key Process Areas related to Requirements Engineering in the CMMI model are:

- REQM Requirements Management
- RD Requirements Development

The Requirements Engineering process itself can be regarded as a means for development process improvement as effective Requirements Engineering speeds up the implementation efforts by eliminating time wasted for clarifying and explaining requirements, as well as reducing the number of defects caused by errors in requirements specification or/and misunderstandings of the requirements documentation due to its low quality. Good Requirements Engineering process increases the quality of testing by providing well described, precise and testable requirements and makes the acceptance testing easier and more reliable as the basis for such testing is usually the requirements.

For training companies: demonstrate the typical requirements for Requirements Engineering process in process maturity models

7 Tool Support	80 minutes

Terms

Change Management Tool, Defect Management Tool, Modeling Tool, Pilot Project, Project Management Tool, Proof of Concept, Prototyping Tool, Requirements Elicitation Tool, Requirements Management Tool, Categories of Tools

Learning Objectives for Foundation Level of Requirements Engineering

The objectives identify what you will be able to demonstrate following the completion of each module.

7.1 Advantages of Tools

LO-7.1.1 Explain the advantages of using tools to support for various activities in Requirements Engineering (K2)

7.2 Categories of Tools

LO-7.2.1 Understand the purpose and usage of different categories of tools supporting Requirements Engineering (K2)

7.3 Selecting Tools

LO-7.3.1 Explain important factors to be considered when selecting a tool for Requirements Engineering (K2)

LO-7.3.2 Explain the process of introducing a tool in an organization (K2)

7.1 Advantages of Tools	30 minutes

There are different tools supporting specific activities of Requirements Engineering.

A common application of tools is to facilitate the storage and administration of requirements. Most requirements management tools provide a common repository for requirements, where requirements can be organized and traceability maintained. It is thus possible to keep complex and changing documents consistent and up-to-date. These tools can automate some mechanical activities and can provide different statistics and reports as well. Modeling tools model requirements in specific modeling notations, such as UML or BPMN. Very often modeling tools can generate the requirements specifications from the models.

In general, tools may support the following activities in Requirements Engineering:

- Requirements elicitation and storage
- Defining and maintaining requirements traceability
- Requirements and solution modeling (including prototyping and business modeling)
- Requirements documenting (requirements specification creating)
- Status reporting
- Configuration and Change Management

Some advantages of using tools are the following:

- Ensuring that all requirements are stored in one place and accessible for all involved stakeholders
- Supporting requirements traceability (e.g., to test cases,) which can be used to verify the relevant requirements coverage
- Managing requirements changes in an easy way that includes configuration control
- Improving the quality of requirements specification by forcing usage of defined document templates and modeling notation
- Saving time by automating some activities (such as generating complete specifications from the tool).
- Providing information for decision making (for example, metrics indicating the number of implemented requirements).

7.2 Categories of Tools	20 minutes

Tools supporting Requirements Engineering activities can be classified as follows:

- Requirements Management Tools
- Requirements Elicitation Tools
- Requirements Modeling Tools
- Prototyping Tools
- Defect Management Tools
- Configuration and Change Management Tools
- Project Management Tools

Many tools facilitate more than one activity, for example modeling tools can offer a requirements repository with configuration and change management facilities, support different modeling notations, documentation generation, and statistics.

For training companies: present an example of a tool supporting at least two Requirements Engineering activities and provide some practical examples of application.

7.3 Selecting Tools	30 minutes

The selection of a tool must occur before the product is developed to avoid the problems that can occur when changing tools mid-project.

The cost of purchasing tools varies greatly. Commercial tools can be very expensive while open source tools are free. The choice of a tool must be made very carefully. Before selecting a tool, there should be an analysis conducted. The analysis might consider the following issues:

- Assessment of organizational maturity, strengths and weaknesses and identification of opportunities for an improved Requirements Engineering process supported by tools
- Requirements regarding the features required from the tool
- A proof-of-concept, by using a test tool during the evaluation phase with the goal to establish whether the tool works effectively with the given Requirements Engineering process and within the current infrastructure and to identify potential changes needed to the existing infrastructure
- Estimation of an ROI based on a concrete business case which can include analyzing the cost of the tool for one specific project or for all/most projects
- Integration between a Requirements Engineering tool and other necessary tools (e.g., defect management tool, project management tool)
- The need for communication between a Requirements Engineering tool and a tool used by a customer organization (in some cases the customer conducts their own Requirements Analysis and those results would then be migrated to the supplier's environment
- Ease of use and learning (potential cost of training), availability of online help, manuals, tutorials, and other support

Introducing the selected tool into an organization should start with a pilot project. Common objectives of a pilot project can be the following:

- Learn more detail about the tool
- Evaluate how the tool fits with existing processes and practices, and determine what would need to change
- Decide on standard ways of using, managing, storing and maintaining the tool and the Requirements Engineering assets

Pilot projects lower the risk of introducing tools, which – if done hastily and without proper analysis – can result in high costs. The following are examples of costly tool issues:

- Selecting a tool that does not meet stakeholder requirements and does not meet its purpose
- Purchasing an expensive tool that is used only on one project or purchasing an expensive tool while there are open source tools with similar functionality
- Required training for a tool without a sufficient help system (especially when only basic functionality is required from the tool)
- Extending the selected tool with additional functions not supported by the tool (while there were other tools with such functionality)
- Integrating the tool with other tools used in the organization

General success factors for the deployment of the tool are the following:

- Adapting and improving processes to fit with the use of the tool
- Rolling out the tool to the organization incrementally
- Providing training and coaching/mentoring for new users, including defining usage guidelines
- Implementing a way to gather usage information from the actual use (including metrics and lessons learned)
- Monitoring tool use and benefits

8 Tables/Pictures

Tables

Figures

9 Standards

IEEE 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology

IEEE 830-1998 IEEE Recommended Practice for Software Requirements Specifications

IEEE 1233-1998 IEEE Guide for Developing System Requirements Specifications

IEEE 1362-1998 IEEE Guide for Information Technology-System Definition – Concept of Operations (ConOps) Document

IEEE 830, IEEE 1233 and IEEE 1362 are replaced by ISO/IEC/IEEE 29148:2011

ISO 9000 Quality management

ISO 12207 Systems and software engineering — Software life cycle processes

ISO 15288 System Life Cycle Processes

ISO 15504 Information technology — Process assessment, also known as SPICE (Software Process Improvement and Capability Determination),

ISO 31000 Risk Management - Principles and Guidelines on Implementation

ISO/EIC 25000 (previously ISO/IEC 9126 Software engineering — Product quality)

ISO/IEC 14598-1:1999 Information technology -- Software product evaluation

SWEBOK - The Guide to the Software Engineering Body of Knowledge: http://www.computer.org/portal/web/swebok/home

SEBOK – Systems Engineering Body of Knowledge: http://www.sebokwiki.org/

10 Books and publications

Cockburn, A.: *Writing Effective Use Cases*. Amsterdam, 2000

Cohn M.: *Estimating With Use Case Points*. Fall 2005 issue of Methods & Tools

Cohn M.: *User Stories Applied: For Agile Software Development.* Addison-Wesley Professional, 2004

Cotterell, M. and Hughes, B.: *Software Project Management*. International Thomson Publishing, 1995

Davis, A. M.: *Just Enough Requirements Management. Where Software Development Meets Marketing*. Dorset House, 2005, ISBN 0932633641

DeMarco, T.: *Controlling Software Projects: Management, Measurement and Estimates*. Prentice Hall, 1986

Doran, G. T.: *There's a S.M.A.R.T. way to write management's goals and objectives*. Management Review, Volume 70, Issue 11, 1981, (AMA FORUM), pp. 35-36.

Evans, E. J.: *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Amsterdam, 2003

Graham, D. et al: *Foundations of Software Testing*. London, 2007

Gilb, T.; Graham, D.: *Software Inspection*. Reading, MA, 1993

Gilb, T.: *What's Wrong with Requirements Specification.* See: www.gilb.com

Hull, E. et al: *Requirements Engineering*. Oxford, 2005

IBAQB: *Certified Business Analyst, Foundation Level Syllabus*, version 2011

IBUQ: *Certified Professional for Usability Engineering*, *Foundation Level Syllabus*, version 2011

ISTQB: *ISTQB Glossary of Testing Terms* v2.2

ISTQB: *Certified Tester, Foundation Level Syllabus*, version 2011

Jacobson, I. et al.: *Object-Oriented Software Engineering. A Use Case Driven Approach*. Addison-Wesley, 1993

Lauesen, S.: *Software Requirements: Styles and Techniques*. London, 2002

Pfleeger, S.L. and J.M. Atlee: *Software Engineering: Theory and Practice.* Upper Saddle River, New Jersey, USA, Prentice Hall, 2006.

Project Management Institute: *A Guide to the Project Management Body of Knowledge* (PMBOK ® Guide). PMI, 2004

Robertson, S.; Robertson, J.: *Mastering the Requirements Process*. Harlow, 1999

Rupp, C.: *Requirements-Engineering und Management. Professionelle, Iterative Anforderungsanalyse in der Praxis*. Munich, 2007

Sharp H., Finkelstein A. and Galal G.: *Stakeholder Identification in the Requirements Engineering Process.* 1999

Sommerville, I.: *Requirements Engineering*. West Sussex, 2004

Sommerville, I.: *Software Engineering 8*. Harlow, 2007

Sommerville, I.; Sawyer, P.: *Requirements Engineering: A Good Practice Guide*. Chichester, 1997

Sommerville, I.; Kotonya, G.: *Requirements Engineering: Processes and Techniques*. Chichester, 1998

Thayer, R. H.; Dorfman, M.: *Software Requirements Engineering, 2nd edition*. Los Alamitos, CA, 1997

Wiegers, K.E.: *First Things First: Prioritizing Requirements.* Software Development, September, 1999

Wiegers, K. E.: *Software Requirements*. Redmond, 2005

Young, R. R.: *Effective Requirements Practices*. Addison-Wesley, 2001

11 Index