Syllabus

REQB®
Certified Professional for Requirements Engineering

Advanced Level
Requirements Manager

Version 1.0

2011

Overview of Changes

Version	Date	Comment
1.0	15.03.2011	First version of the syllabus

Main Idea

The central theme for this syllabus was that the complexity of software and our dependency on software continues to increase. The result is a high level of dependency on the freedom from error in the software. 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 now create such a uniform language in this important area of requirements engineering, international experts got together in REQB and developed this syllabus.

Acknowledgements

Requirements Engineering Qualifications Board Working Party Advanced Level (Edition 2011): (Karolina Zmitrowicz (chair), Krzysztof Chytła, Eric riou du Cosquer, Michał Figarski, Olivier Narat, Folke Nilsson, Bartłomiej Prędki, Alain Ribault, Robert Treffny)

Table of contents

Introduction

Purpose of the Syllabus

This syllabus defines the Advanced Level of the training program to become a REQB Certified Professional for Requirements Engineering (short form CPRE) in the area of Requirements Management. REQB developed this syllabus in cooperation with the Global Association for Software Quality.

The syllabus is to serve as a foundation for training providers who are seeking accreditation as teachers. All areas of this syllabus must correspondingly be incorporated in the training documents. The syllabus should, however, also serve the learner as preparation material for certification. All areas listed here are thus relevant for the examination, which can be taken either after completion of an accredited courses or independently in open examination.

Examination

The examination to become a Certified Professional for Requirements Engineering Advanced Level, Requirements Management is based on this syllabus. All sections of this syllabus can thereby be tested. The examination questions are not necessarily divided into the individual sections. A question may refer to several sections.

The format of the examination is Multiple Choice.

Examinations can be taken after having attended accredited courses or in open examination (without a previous course). You will find detailed information regarding examination times on the website of gasq (www.gasq.org) or on REQB's website (www.reqb.org).

Accreditation

Providers of a REQB Certified Professional for Requirements Engineering Advanced Level Course must be accredited by the Global Association for Software Quality. Their experts review the training provider's documentation for accuracy. 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 several international experts. The content of this syllabus can therefore be seen as an international standard. The syllabus thereby makes it possible to train and examine internationally at the same level.

K Levels

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.
There are 4 K-levels in the current syllabus:

- K1 - remember, recognize, recall
- K2 - understand, explain, give reasons, compare, classify, summarize
- K3 - apply in a specific context
- K4 – analyze

System or Software

This syllabus is using the term System for the scope for this document, but that word can in this context means a plain software product or solution or it can be a combination of Hardware, software, documentation, services connected to it.

1. Basics (K2) — 60 minutes

Learning Objectives for Advanced Level of Requirements Engineering

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

1.1 Repeating basics (K2)

LO-1.1.1	Recall what a requirement, Requirements Engineering, Requirements Development and Requirements Management are (K1)
LO-1.1.2	Explain with examples what the meaning and purpose of requirements are (K2)
LO-1.1.3	Explain how requirements can be classified (K2)
LO-1.1.4	Explain with examples what problems may be related with requirements (K2)

1.1 Repeating basics (K2) | 60 minutes

Terms

Functional Requirement, Non-functional Requirement, Requirement, Requirements Development, Requirements Engineering, Requirements Management, Process Requirements, Product Requirements

Background

Requirement [After IEEE 610]

- A condition or capability needed by a user to solve a problem or achieve an objective.
- 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.
- A documented representation of a condition or capability as in (1) or (2).

Requirements serve as a foundation for further software development. The meaning of requirements can be expressed as (K2):

- Foundation for assessment, planning, execution and monitoring of the project activity
- Customer expectations
- Component of agreements, orders, project plans
- Setting of system boundaries, scope of delivery, contractual services

Classification of requirements [Ebert] (K2)

Requirements may be related not only to software system, but to business process or organizational structure as well. For instance, the purpose of project may be not only implementing software components but also improving the organization's business process itself.

Requirements can be divided into process requirements and product requirements.

Process requirements express expectations and needs related to costs, marketing, processing time, sales and distribution, organization, documentation. They also describe needs and limitations of the business processes. For example, process requirements may specify methodologies to be followed and the constraints that an organization must respect.

Product requirements consist of functional and non-functional requirements. Both can be regarded from the point of view of a user (external) or a customer and from the point of view of a development team (internal).

- Functional product requirements allow to specify what the product should do. Functional requirements describe the function of the product.
- Non-functional product requirements specify how the system performs its functions. Non-functional requirements describe the quality attributes of the system.

A product is defined as a composition of software, hardware and other outputs of the production process (here Software Development process) like documentation and code.

<u>Requirements Engineering (K2)</u>

Requirements Engineering is a sub discipline of software engineering, focused on determining and managing the requirements of hardware and software systems. Requirements Engineering discipline involves the following sub-processes: requirements elicitation, analysis and negotiation, specification, system modeling, requirements validation and requirements management.

<u>Requirements Development (K2)</u>

Collection of activities, tasks, techniques and tools to identify, analyze and validate requirements. Includes the process of transforming needs into requirements.

<u>Requirements Management (K2)</u>

A continuous process of eliciting, documenting, analyzing, tracing, prioritizing, communicating. agreeing on requirements and managing requirements' changes.

Identification and analysis of the requirements is a challenging task. The following problems may appear during the Requirements Engineering process (K2)

- Ambiguous, overly specified, unclear, impossible, contradictory requirements
- Unclear objectives of the project
- Communication problems (within or between both vendor and customer organizations)
- Language barriers
- Knowledge barriers (for example, stakeholders representing expertise in different domains , like business, technology etc.)
- Vague formulation
- Too formal description
- Instability of the requirements (example: changing business environment)

- Bad quality of the requirements (incomplete, not well described)
- Gold plating (passing through controversial measures)
- Insufficient user involvement
- Overlooked user classes (missing stakeholders groups)
- Inaccurate planning
- Minimal specification

To reduce the risk of having problems with requirements and to ensure the best possible quality of the future software, the following quality criteria for requirements should be [Wiegers] (K2):

- Each requirement must be complete, correct, feasible, necessary, prioritized, unambiguous, and verifiable
- The requirements specification must be complete, consistent, modifiable and traceable

One of the main purposes of requirements is to express stakeholders' expectations and needs regarding the planned software. Requirements must express real value understood as perceived benefit for the stakeholders [Gilb].

<u>Requirement Manager (K2)</u>

A Requirement Manager is a person responsible for documenting, analyzing, tracing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders.

<u>Requirement Developer (K2)</u>

A Requirement Developer is a technical oriented person mainly involved in the Elicitation, Analysis and prioritizing of requirements.

2. Processes models and management (K4) | 180 minutes

Learning Objectives for Advanced Level of Requirements Engineering
The objectives identify what you will be able to do following the completion of each module.

2.1 Process models (K3)

LO-2.1.1	Analyze a software project and determine the process model on which it is based (K4)
LO-2.1.2	Apply the Agile principles to a software project (K3)
LO-2.1.3	Analyze a scenario showing how to use Requirements Engineering in Agile techniques (K4)
LO-2.1.4	Compare different approaches to Requirements Engineering (K2)

2.2 Management and control of the Requirements Engineering Process (K3)

LO-2.2.1	Apply solutions and tools to manage and control the Requirements Engineering process (K3)
LO-2.2.2	Analyze a scenario describing requirements and determine which point of view is expressed (K4)

2.1 Process Models (K3) | 100 minutes

Terms

Agile Manifesto, Agile Model, Crystal Clear, Extreme Programming (XP), Heavyweight Model Incremental Model, Iterative Model, Lightweight Model, Product Backlog, Rational Unified Process (RUP), Scrum, Software Process Model, Sprint, Sprint Backlog, User Story, Waterfall Model, V-Model,

Background

A software process model gives a standard format for planning, organizing and running a software project.

There are traditional software process models proposing various approaches to develop software products: Waterfall Model V-Model, Rational Unified Process (RUP) etc. The main drawback is that some of these models have a more complex method for control of changing requirements.

Requirements Manager should know how to manage requirements in different process models.

Waterfall model (K3)

The waterfall model is an early model having its origins in the manufacturing and construction industries (Royce 1970). It is a sequential model with the following phases:

Requirements specification
→ Design specification
→ Implementation
→ Verification
→ Maintenance

Its major disadvantage is that it is difficult to change requirements after the specification phase is completed.

Thus, the waterfall model should only be used in the following situation:

- The requirements are clear and not subject to change.

- The software development organization has experience on similar projects.
- It is important to complete each phase before starting the next one (e.g. to have clearly and precisely defined requirements before moving to the design specification step).

V-Model (K3)

The V-Model was developed as an extension of the waterfall model in the late 1980s. It has a V-shape and each phase has an associated testing phase:

Requirements Analysis	→	User Acceptance Testing
System Design	→	System Testing
Architecture Design	→	Integration Testing
Module Design	→	Unit Testing
	Coding	

Since there is an early involvement of testing, defects can be detected earlier and testers can develop a better understanding of the product. This model is often a base for more iterating and agile approaches.

Rational Unified Process (K3)

The Rational Unified Process (RUP) was originally developed in 1998 by Rational Software (acquired by IBM in 2003). It is an iterative model (i.e. the process is repeated until all scenarios, risks and changes have been addressed. It has 9 disciplines including a requirements discipline (6 engineering disciplines plus 3 supporting disciplines). Each discipline is covered by 4 consecutive project life-cycle phases: inception, elaboration, construction, transition.

With the Rational Unified Process, the requirements are collected and specified the following way:

- The Business Modeling discipline establishes a basic knowledge of the problem domain with initial Use Cases, Business Use Cases and Business Actors. Its goal is to ensure a common understanding between different stakeholders like: customers, end users, developers.
- The Requirements discipline focuses on Use Cases to capture requirements. It has the following activities:
    - Analyze the Problem: The key stakeholders and key requirements are identified
    - Understand Stakeholder Needs: More detailed requirements are collected with various techniques: customer interviews, Story Boards etc.

- Define the System: System Actors and Use Cases are identified and refined
- Manage the Scope of the System: the boundaries of the system are analyzed; Use Cases are prioritized
- Refine the System Definition
- Manage changing requirements

The Rational Unified Process has the following advantages:

- It is a documented framework
- It addresses changing requirements
- It can be adjusted to the specific needs of a project: only what is necessary is taken from the RUP framework, nothing more

<u>Agile models (K3)</u>

In contrast with the traditional "heavyweight" models, there are "lightweight" models (agile models), which support changing requirements: Scrum, Crystal Clear, Extreme Programming (XP), etc. Agile models were started by the Agile Manifesto in 2001. They are iterative as well as incremental models (i.e. the product is designed, developed and tested in small portions at a time).

The Agile Manifesto promotes:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

"Welcome changing requirements, even late in development" is one of the 12 principles behind the Agile Manifesto.

Requirements are created by questioning customer representatives. Their expectations are written down on post-it notes and captured into User Stories. A typical scenario of Requirements Engineering in Agile techniques is the following:

- One of the developers meets customer representatives
- The developer asks questions about a specific feature (e.g. the display of products on an e-commerce website)
- The User Stories are formulated by the customer representatives

- The User Stories are written down on note cards (e.g. “As a user, I want to see several images of the products”)
- The User Stories are prioritized by the customer
- The developers start to implement the User Stories. They may ask the customer representative for clarification in every time

The Agile approaches are useful in cases where the “customer” doesn’t know his requirements and thereby there are problems in making reliable estimation of efforts. Agile approach supports frequent changes in requirements and reprioritizing

Scrum (K3)

Scrum was published in 1995. It has predefined roles (“Scrum Master”, “Product Owner”, “Team”) and practices (time-box periods of development called “sprints”). It uses a prioritized list of high level requirements (“product backlog”) and a list of tasks to address during a “sprint” (“sprint backlog”) [http://www.scrum.org/scrumguides].

With Scrum, changing requirements are managed the following way:

- New requirements are put to the Product Backlog.
- The Sprint Backlog is changed to include new requirements (tasks) with high priorities. Some of lower priority tasks are moved to the next sprint.

Scrum is one of methods for planning and controlling the Agile type of development and an alternative to traditional project management methods.

Crystal Clear (K3)

Crystal Clear is an agile model that focuses on people (small team of less than 10 people). It is characterized by frequent deliveries and communication with users. Additionally, it can include the following characteristics: personal safety, focus, easy access to expert users, automated tests.

Crystal Clear is well-suited for non-safety critical projects with changing requirements.

Extreme Programming (XP) (K3)

Extreme Programming (XP) started being adopted in the late 1990s and early 2000s. It uses 4 basic activities: coding, testing, listening, and designing. It has 5 values: communication, simplicity, feedback, courage, respect. It also has 12 practices derived from the best practices of software engineering and grouped into four areas (“planning game” among others where high-value requirements are written down on “user story” cards).

2.2 Management and control of the Requirements Engineering Process (K3)	80 minutes

Terms

Requirements Engineering, Identification of Requirements, Requirements Analysis, Specification of Requirements, Tracking and Control of Requirements, Quality Assurance, Internal Side, External Side, UML/SysML, Use Cases, Stakeholder, Traceability

Background

Requirements Engineering is a systematic requirements process (management and development). It is a key activity undertaken in order to identify the needs of various stakeholders and treat them in a structured way.

The Requirements Engineering process (refer to Chapters 4 – 8 for more details) can be divided into the following chronological steps:

- Identification of Requirements
- Specification and Documentation of Requirements
- Requirements Analysis
- Tracking and Control of Requirements
- Quality Assurance of requirements management and development

There are various stakeholders with different points of view about requirements. In general, requirements can be regarded from the point of view of the user (external) or customer and from the point of view of the vendor's team (internal).

Examples:

- Functional product requirements from the user's or customer's point of view: user interface, applications, services
- Functional product requirements from the vendor's team (i.e. developer) point of view: architecture, power supply, load distribution
- Non-functional product requirements from the point of view of the user: performance, usability
- Non-functional product requirements from the point of view of the customer: reliability, performance

- Non-functional product requirements from the point of view of the vendor’s team: testability, serviceability, tools

For more details how to identify stakeholders refer to chapter 4.2 Identifying Stakeholders (K2).

Different factors influence Requirements Engineering: Organization’s maturity and culture, regulations, etc. Refer to chapter 4.6 Factors affecting Requirements Engineering (K2) for more details.

Some factors may have a negative influence on requirements engineering:

- On the internal side:
    - Lack of knowledge of the user’s domain
    - Ineffective Requirements Engineering approach/methodology
    - Insufficient personnel experience and skill
- On the external side:
    - Lack of communication
    - Unclear and/or changing business objectives
    - No knowledge about the software development process
    - No involvement of users

Possible solutions are:

- Workshop techniques, Requirement camps etc,
- Using the Unified Modeling Language (UML)/Systems Modeling Language (SysML) to visualize requirements
- Using a Domain Specific Language to have a common vocabulary between stakeholders (users, project managers, software providers, developers, etc.)
- Using the advantages of Agile methods to handle requirements: collaboration with the customer, User Stories in everyday language, Sprint Backlog showing a list of tasks to address
- Tools to track and control requirements

3. Project and Risk Management (K3) — 240 minutes

Learning Objectives for Advanced Level of Requirements Engineering

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

3.1 Project Management (K2)

LO-3.1.1	Explain why Requirements Engineering is necessary in IT projects (K2)
LO-3.1.2	Describe the project areas affected by Requirements Engineering and the main products of Requirements Engineering in specific phases of the project (K2)
LO-3.1.3	Explain with examples the impact of Requirements Engineering and its outputs on project planning (K2)

3.2 Risk Management (K3)

LO-3.2.1	Explain with examples what risks can appear within the project (K2)
LO-3.2.2	Describe the process of Risk Management (K3)
LO-3.2.3	Explain and use different techniques and approaches to risk identification (K3)
LO-3.2.4	Describe how risk can be assessed (K3)
LO-3.2.5	Explain and provide examples of possible techniques to manage the risk (K3)
LO-3.2.6	Describe the meaning and content of Risk Management Plan (K2)
LO-3.2.7	Provide examples of requirements-related risks (K3)
LO-3.2.8	Explain and use in given scenario the process of developing FMEA (K3)
LO-3.2.9	Recall the advantages of using FMEA (K1)
LO-3.2.10	Recall the common techniques of Risk Reduction (K1)
LO-3.2.11	Explain how prototyping can help with Risk Reduction (K3)

3.3 Roles in Requirements Management (K2)

LO-3.3.1	Describe the common roles in Requirements Management (K2)

3.1 Project Management (K2) — 30 minutes

Terms

Contract, Project Management, Requirement Management Plan

Background

Problems with requirements are one of the main reasons why projects fail. Neglecting Requirements Engineering can lead to the following issues:

- Requirements are not precise
- Requirements are contradictory
- Requirements that change during the software development (does not apply to Agile approaches)
- Requirements do not fulfill the criteria and do not satisfy stakeholders needs
- Requirements that can be interpreted differently by different readers
- Missing requirements

Therefore Requirements Engineering is necessary in IT projects and should be included in the project life cycle and carefully planned (K2).

Requirements Engineering should contribute to the following areas (K1):

- Project conception
- Contract negotiations
- Project definition
- Project execution

<u>Project conception</u>

The role of Requirements Engineering in project conception area is to identify customer's needs and expectations regarding the solution of the problem and to establish high-level requirements that will be base for project estimating, scope planning and further design.

<u>Contract negotiations</u>

In contract negotiations step, products of Requirements Engineering allow to determine the scope and required resources for the project as well as establish costs of development. The important task when negotiating contract is to agree on priorities of requirements.

Project definition

Requirements Engineering (RE) is a main base for Project definition. It allows to create a detailed business design of the solution and contribute to architecture design. Products of the Requirements Engineering are a base for preparing test cases and test scenarios and further testing of the solution (as requirements specification together with other documents created on the base of requirements specification like functional specification, is a test basis). It is important to notice that Requirements Engineering has significant impact on planning, as the outputs of Requirements Engineering determine the scope of the system and project.

The Requirements Engineering contributes to:

- Project Plan
- Quality Plan
- Detailed Analysis and Design Plan
- Development Plan
- Testing Plan

The Quality Plan may contain other plans and, from the other side, may be a part of other plans. For example, the Quality Plan may describe the processes used for Requirement Engineering and can be further detailed in a Requirement Management Plan.

Project execution

During Project execution, products of Requirements Engineering are a base for requirements development and requirements verification (testing).

Any changes in requirements should lead to reviewing the proper plans and adjusting them to the current scope of the solution.

Projects are of many different kinds and lengths and they need different type of Requirement Engineering approaches. These approached can be grouped in to some general categories.

- Elephants (Heavy, long-lived and with good memory) They often need a complete Requirement documentation involving many parties, often localized in different places
- Horses (Stable, strong and independent) The most common type of development project which need certain type of formalization and control of Requirements
- Rabbits (The fastest and most Agile) Where development is made in short iterations and requirements are often expressed in terms of prototypes or sceneries

Requirements Management Plan (K2)

Requirements Management activities should be documented in a form of the Requirements Management Plan. Template of the Requirements Management Plan includes the following information:

1. Overview

2. Objectives

3. Identification of Requirements Scope

4. Roles and Responsibilities

    4.1. Requirements Engineer or Developer
    4.2. Configuration Manager (CM)
    4.3. Customer
    4.4. QA/Test Manager
    4.5. Configuration (Change) Control Board (CCB)

5. Processes and Procedures

    5.1. Identification of Requirements
    5.2. Recording Requirements
    5.3. Change Management
    5.3. Modification of Requirements
    5.5. Verification and Validation

6. Requirements Management Tools

    6.1. Requirements Tracking Matrix

7. Schedule

3.2 Risk Management (K3) | 180 minutes

Terms

Failure Mode and Effects Analysis, Product Risk, Project Risk, Prototyping, Risk Analysis, Risk Avoidance, Risk Identification, Risk, Risk Management, Risk Management Plan, Risk Mitigation, Risk Reduction, Risk Retention, Risk Sharing

3.2.1 Fundamentals of Risk Management (K3)

<u>Risk (K1)</u>

Risk is defined as the effect of uncertainty on objectives, whether positive or negative [ISO 31000].

The other definition describes risk as the chance of an event, hazard, threat or situation occurring and resulting in undesirable consequences or a potential problem. The level of risk is determined by the likelihood of an adverse event happening and the impact (the harm resulting from that event).

<u>Types of risk (K2)</u>

There are two types of risk (K2):

- Product risk
- Project risk

<u>Project risks (K2)</u>

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

- Organizational factors:
  - Skill, training and staff shortages
  - Personnel issues
  - Political issues, such as:
    - Problems with stakeholders communicating their needs and expectations
    - Failure to follow up on information found in reviews (example: not improving requirements documenting practices)
  - Improper attitude toward or expectations of Requirements Engineering

- Technical issues:
    - Problems in defining the right requirements
    - The extent to which requirements cannot be met given existing constraints
    - Development or test environment not ready on time
    - Low quality of the design, code, configuration data, test data and tests
- Supplier issues:
    - Failure of a third party
    - Contractual issues

Product risks (K2)

Potential failure areas (adverse future events or hazards) in the software or system are known as product risks, as they are a risk to the quality of the product. These include:

- Failure-prone software delivered
- Low quality software documentation (incomplete, inconsistent, difficult to maintain)
- The potential that the software/hardware could cause harm to an individual or company
- Poor software characteristics (e.g., functionality, reliability, usability or performance)
- Poor data integrity and quality (e.g., data migration issues, data conversion problems, data transport problems, violation of data standards)
- Software that does not perform its intended functions and does not satisfy the needs of stakeholders
- Business risk based on bad quality

Risk Management (K3)

Risk Management is the process of identification, assessment and prioritization, planning reaction on risks and resolving and monitoring of risks. It is one of the most important processes as it allows to identify potential factors that may have a negative effect on the execution of a project and prepare proper action to deal with the risk if it appears.

Risk Management should involve all stakeholders or representatives of particular groups of stakeholders on the project. Different risks may come from different groups of stakeholders: for example, the development team will see different risks than business stakeholders or end-users.

Risk identification (K3)

Risk identification can be conducted in two ways: starting with the source of problems, or with the problem itself.

- Source analysis – the sources of risk may be internal or external to the project. This approach to risk identification requires detecting all potential sources of a problem (stakeholders, employees of a company, external regulations, technology) and then identifying risks associated with these sources.
- Problem analysis – in this approach risks are related to identified threats. It requires detecting all potential problems that may appear with the software or project (like: losing money, abuse of privacy information, accidents) and then identifying risks associated with these problems.

<u>Risk identification methods (K3):</u>

There are several methods of identifying risks. Selecting a method to be used on the specific project may depend on organizational culture, domain, experience or industry practice. Common risk identification methods are:

- Objectives-based risk identification
- Scenario-based risk identification
- Taxonomy-based risk
- Common-risk checking
- Risk charting
- Expert interviews
- Estimation of independent consultants
- Experience (lessons learned)
- Checklists for typical risk (risk templates)
- Risk workshops
- Brainstorming

<u>Risk analysis (K3)</u>

As the risks have been identified, they must be assessed as to their potential severity of loss and to the probability of occurrence. Risk analysis is the study of identified risks, specifically categorizing each risk and establishing associated with it probability and impact. Risks are categorized by allocating them into an appropriate type. Typical quality risk types are defined in ISO 25000 standard.

The probability of risk occurrence may be expressed as a percentage (1-99%) or as a level (example: low-high). Similarly, the consequences (severity) of the risk occurrence may be expressed as a percentage or as a level (example: no, minor, major).

The assessment should provide the best possible guesses in order to properly prioritize the risks and create realistic Risk Management Plan.

The most common formulas for expressing risk are (K3):

- Rate of occurrence multiplied by the impact of the event equals risk
- Composite Risk Index (Composite Risk Index = Impact of Risk event x Probability of Occurrence)

The most important problem related to risk assessment is determining the rate of occurrence as reliable statistical information is often not available. Another problem is assessing the severity of the risk impact. This value is difficult to evaluate for immaterial assets.

Planning reaction on risk (K3)

Planning reaction on risk (Risk mitigation) should answer the question: what should be done if the risk appears. It may include preparation of an emergency plan and preparation of potential solutions.

Potential risk treatments (K3)

Techniques to manage the risk can be divided into four major categories [Dorfman, Mark S.]:

- Avoidance
- Reduction
- Sharing
- Retention

As a result of risk identification, analysis and planning adequate reactions, there should be a Risk Management Plan created.

Risk Management Plan (K3)

Risk Management Plan should be created before and after (periodically updated, for example after each iteration) creation of the Project Plan as some of activities defined for mitigating the risk may affect the Project Plan and – from the other side – some of project tasks will cause additional risks or will change the attributes of already detected risks.

Risk Management Plan includes:

- List of risks
- Probability of occurrence and/or priority
- Severity of impact for each of risks (including cost if applicable)
- Mitigation strategies for each of risks (including the person/group responsible for taking actions)
- Risk assessment matrix

The Risk Management Plan should provide effective security controls for managing the risks and contain a schedule for control implementation and responsible persons for those actions.

<u>Characteristics of Risk Management (K1)</u>

Risk management should [ISO 31000]:

- Create value
- Be an integral part of organizational processes
- Be part of decision making
- Explicitly address uncertainty
- Be systematic and structured
- Be based on the best available information
- Be tailored
- Take into account human factors
- Be transparent and inclusive
- Be dynamic, iterative and responsive to change
- Be capable of continual improvement and enhancement

<u>Risk resolving and monitoring (K3)</u>

Having a plan to deal with the risk does not solve the risk. When the risk appears, then previously defined and planned mitigation actions should be implemented. Efficiency of dealing with the risk should be monitored in order to achieve better result in the future.

Risk analysis results and plans should be systematically updated. The reasons for this are the following:

- To evaluate whether the previously defined risk mitigation strategies are still applicable and effective
- To evaluate the possible risk level changes in the business or project environment
- To find out if any new risks appeared

It is important to systematically report Risks and mitigations and perform consecutive risk analysis, for example in large projects and in case of Gate passing.

Failure Mode and Effects Analysis (K3)

Common technique for Risk management (identification, analysis and planning reaction) is Failure mode and effects analysis (FMEA).

FMEA allows to prioritize failures according to the severity of consequences, frequency of the occurrence and how easily they can be detected. FMEA also documents current knowledge and actions about the risks of failures for use in continuous improvement. In most cases FMEA is used during the design stage of the project and its main purpose is to avoid future failures. In later phases it can be used for process control. FMEA should begin in the earliest conceptual stages of the project and continue throughout the whole life cycle. Ideally, FMEA should be scheduled as soon as the preliminary information is available.

The results of an FMEA are actions preventing or reducing the severity or likelihood of failures.

<u>FMEA implementation steps (K3)</u>

FMEA is developed in three main phases:

- Step 1: Severity
- Step 2: Occurrence
- Step 3: Detection

Like a Risk Management Plan, FMEA should be also systematically updated. The reasons or triggers for an update may be:

- Starting new design / development cycle or introducing new product or process
- Requesting changes to the operating conditions
- Making changes in the requirements specification or solution design
- Instituting new regulations
- Receiving customer feedback indicating a problem

<u>Advantages of using FMEA (K1)</u>

Advantages of using FMEA may be the following:

- Improving the quality of a product/process
- Increasing stakeholders' satisfaction by eliminating common mistakes and improving the overall quality of the final product
- Gathering information allowing to avoid or reduce future failures

- Collecting engineering knowledge what can be then used for creation organizational best practices
- Early identification and elimination of potential failure modes
- Minimizing late changes and associated cost
- Minimizing the probability of similar failures in future

Disadvantages of using FMEA (K1)

- High costs
- Work effort
- Extended documentation
- Early involvement

3.2.2 Requirements-related risks (K3)

Requirements-related risks (K2)

Most common risks related to requirements are:

- Instability of requirements including scope changes – resulting from changing business environment (optimization or changes in business processes, internal procedures and regulations etc.), difficulties in collecting the right requirements from stakeholders (stakeholders may not be able to articulate their requirements in the beginning of the project – as a result, the initial requirements may be found are not correct and incomplete).
- Missing requirements – resulting from ineffective requirements elicitation process (Business Analysis has not collected all necessary requirements what may be caused by various reasons: lack of domain knowledge, ineffective elicitation techniques, no stakeholders' involvement or missing some important stakeholders in requirements elicitation process).
- Low quality of requirements specification – what may be caused by ineffective Requirements Analysis and Documentation processes, tight schedule or no standards related to requirements specification within an organization. As a consequence of low quality of requirements specification, other areas of project will be also affected – for example implementation and testing will take more time, as the team will need to clarify and detail unclear parts of the specification.

- Requirements not providing the real value [TGilb] – requirements describe functionality and solutions but no real value of what has to be delivered to the stakeholders. The reason of that may be lack of understanding of the business goals of the project (like: increasing sales to defined amount).
- Communication – in many cases there are serious problems with communicating the requirements to the other members of the project team. Requirements should be known to all stakeholders, any changes done in the requirements specification must be also communicated to the team. This is often neglected. In result, some groups of stakeholders may not be aware of the current status of the requirements.

<u>Mitigating requirements-related risks (K3)</u>

Requirements-related risks can be mitigated using the following safety means:

- Adopting effective requirements elicitation techniques (brainstorming, workshops, prototyping).
- Adopting standards and best practices related to Requirements Engineering and adjusting them to an organization's and specific project needs.
- Creating or adopting requirement specification templates what forces using specific form of documenting the requirements.
- Introducing reviews and audits as a part of Quality Assurance activities.
- Change Management as a process to manage any changes in requirements.
- Establishing Communication Plan describing the process of requirements communication together with the frame schedule and responsibilities.
- Instructing the whole team about introduced rules and procedures applying to the Requirements Management and ensuring they all understand and will apply them.

3.2.3 Risk Reduction through Prototyping (K3)

<u>Prototyping (K1)</u>

In Software Development, prototyping is the activity of creating prototypes of software application, its components or modules or single screens. A prototype typically simulates only a few aspects of the final solution (most often those of the highest level of uncertainty or risk).

The benefits of prototyping may be the following (K2):

- Help in identifying and detailing requirements (especially when stakeholders are not able to clearly state their needs and expectations).
- Help in requirements documentation – prototyping not only makes the documenting process easier (as the writer can base the document on visible and testable prototype: this is especially useful in describing user interface requirements and aspects related to the navigation), but allows to verify the correctness and ability to implementation of already defined requirements. Furthermore, when using prototyping very often new requirements are discovered, not identified in initial requirements elicitation process.
- Requirements Engineer and other members of development team can get valuable feedback from the stakeholders early in the project.
- Stakeholders may check if the proposed solution satisfies their needs and expectations.
- Prototyping allows to verify the accuracy of initial project estimates as it gives some insight into the work to be done.

3.3 Roles in Requirements Management (K3) | 30 minutes

Terms

CCB, Configuration Manager, System Analyst

Background

There are various roles in Requirements Management. Each of the roles should be listed in Requirements Management Plan.

<u>Customer</u>

The customer is responsible for defining and approving all requirements, and all modification to requirements. In Requirement Identification activities, the customer is usually supported by Business and System Analysts and Requirements Engineers. It is also important to look at customers in a wider scope like Stakeholders and Reference groups.

<u>Project Manager</u>

A Project Manager is concerned with planning, budgeting, resourcing and scheduling the project. Knowledge of the requirements is essential for detailed planning and estimation purposes.

<u>System Analyst</u>

A System Analyst details business requirements developed by a Business Analyst and a Requirement Engineer, analyzes integration requirements, plans implementation solutions and coordinates development to meet business and other requirements. The System Analyst should be familiar with multiple approaches to problem-solving. Analysts are often familiar with a variety of programming languages, operating systems, and computer hardware platforms. The System Analyst often write user requirements into technical specifications (like Use Case specifications based on business requirements), linking business requirements with requirements resulted from the current IT infrastructure within the organization, technology to be applied etc.

The term System Analyst and Requirements Engineer are often used to define the same role within the project. However, the main difference between the two terms is that the Requirements Engineer's role is to elicit, analyze and develop system requirements to solve a given business problem (defined by a Business Analyst and expressed in a form of business requirements) when the System Analyst designs the solution at the system level – considers the topology, integration interfaces, technical requirements etc.

Requirements Engineer

The Requirements Engineer is responsible for:

- Working with the customer to identify detailed requirements (based on business requirements provided by the Business Analyst)
- Providing requirements models or other visualizations that may facilitate in the communication of the detailed requirements and in ratifying the proposed solution
- Analyzing an impact of customer's requirements or changes in the existing requirements submitted to the Change Control Board
- Communicating requirements impact to the customer
- Negotiating requirements modification when needed

Requirements Developer

The Requirements Developer is responsible for the collection of activities, tasks, techniques and tools to identify, analyze and validate requirements, including the process of transforming needs into requirements.

Requirements Manager

The Requirements Manager is responsible for the overall process of eliciting, documenting, analyzing, tracing, prioritizing, communicating, agreeing on requirements and managing requirements' changes.

Change Control Board (CCB)/Product Control Boards

The CCB is responsible for analyzing and evaluating requirements for feasibility and impact to the project or product.

Configuration Manager (CM)

The CM is responsible for:

- Maintaining a matrix of all customer approved requirements
- Oversight of the requirements change control process
- Applying changes to requirements matrix

- Maintaining the modification history of requirements

QA/Test Manager

The QA/Test Manager is responsible for:

- Verifying that the delivered product satisfies the approved customer's requirements
- Documenting the results of the requirements verification in a Test Analysis Report

Quality Manager

The Quality Manager is responsible for verifying if the organization has the necessary processes, procedures and tools for Requirement Engineering and is proper trained for using them. He/she is responsible for Audits and Measurement of the Requirement Engineering processes/procedures.

4 Identification of Requirements (K4) | 350 minutes

Learning Objectives for Advanced Level of Requirements Engineering
The objectives identify what you will be able to do following the completion of each module.

4.1 Customer (K3)

LO-4.1.1	Describe how the customer may influence the Requirements Identification process (K3)
LO-4.1.2	Explain the role of the customer in the Requirements Identification process (K2)
LO-4.1.3	Explain the techniques of involving the customer in the Requirements Identification process (K3)

4.2 Identifying Stakeholders (K2)

LO-4.2.1	Explain the concept of stakeholder and its role in the project (K2)
LO-4.2.2	Describe how stakeholders can be identified (K2)
LO-4.2.3	Explain and provide examples how expectations of different stakeholders can affect the product vision (K2)

4.3 Techniques for Identifying Requirements (K3)

LO-4.3.1	Describe on examples common techniques for Identifying Requirements (K3)
LO-4.3.2	Use selected techniques for identifying requirements in given scenario (K3)

4.4 Functional and Non-functional Requirements (K3)

LO-4.4.1	Explain what Functional and Non-functional Requirements are (K2)
LO-4.4.2	Explain the meaning of Non-functional Requirements for the stakeholders and their satisfaction from the software product (K3)

4.5 Description of Requirements (K3)

LO-4.5.1	Explain the purpose of describing requirements (K2)
LO-4.5.2	Describe the standard content of requirement document (K2)
LO-4.5.3	Explain with examples common mistakes done in Requirement Documentation (K3)
LO-4.5.4	Describe requirements for a given scenario (K3)

4.6 Factors affecting Requirements Engineering (K2)

LO-4.6.1	Explain and provide examples of how application domain can affect Requirements Engineering (K2)
LO-4.6.2	Explain and provide examples of how organization and culture can affect Requirements Engineering (K2)
LO-4.6.3	Explain and provide examples of how organizational and technical maturity can affect Requirements Engineering (K2)
LO-4.6.4	Explain and provide examples of how combination hardware, software and services can affect Requirements Engineering (K2)

4.7 Differences and similarities between a general product and a specific solution (K3)

LO-4.7.1	Analyze, compare and explain differences and similarities between a general product and a specific solution (K3)

4.1 Customer (K3)	30 minutes

Terms

Client, Customer, Purchaser

Background

Customer – the organization or person purchasing the software is one of the key stakeholders of the project. Customer's needs must be satisfied.

The following is required to achieve success in collecting right and sufficient requirements:

- Communicating effectively with the customer
- Knowing and respecting the customer's needs and expectations
- Knowing the domain of the customer's business

Customer involvement is necessary in the following cases:

- When the software is designed and developed from scratch
- When the ready software product (COST – component-off-the-shelf) is customized according to the needs of specific project (example: by adding new functions, modifying default product's functions)
- When the software is equipped from other company and only deployed into the customer's environment

Customer should provide initial business needs and expectations. Usually it is provided together with the offer/service request. It is the vendor responsibility to explore these needs and extract requirements on that basis. Such initial requirements are a base for estimating the project cost and schedule and should be obligatory documented in the contract with the customer. In addition, for each of such requirements there should be acceptance criteria defined and approved by both sides.

<u>Contract (K2)</u>

Contract should include:

- Short description of the planned solution
- The list of prioritized requirements
- The acceptance criteria for each requirement
- The list of products (documentation, code, working software)

- Deadlines for development and delivery of the product
- Other needs and expectations like preferred technology to be used, resource requirements etc.

The vendor should declare which of listed requirements will be implemented by the project and that the deadlines are accepted. In case of risks related to the requirements or dates, the vendor should discuss the problems with the customer and get consensus on it.

4.2 Identifying Stakeholders (K2) — 40 minutes

Terms

Analyzing functions, Baseline Stakeholder, External Stakeholder, Internal Stakeholder, Questionnaire, Stakeholder, Satellite Stakeholder

Background

Stakeholder is any person or organization who has an interest in an project. Project 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. Stakeholders can exercise control over both the immediate system operational characteristics, as well as over long-term system lifecycle considerations (such as portability, lifecycle costs, environmental considerations, and decommissioning of the system) [after TGilb].

When starting a new project, it is very important to identify all stakeholders as each of them or each of groups 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) may remain unknown. The result of missing user classes may be significant, for example the need of complex changes in the software in late stage of the project or even after releasing the system into production environment.

There may be many categories of stakeholders including:

- End-users
- Managers
- People involved in the organizational processes
- Engineers responsible for system development and maintenance
- Customers of the organization who will use the system
- External bodies (regulators)
- Domain experts

Each of these categories of stakeholders has different needs and goals, sometimes conflicting with the goals or expectations of other stakeholders.

Categories of stakeholders (K2)

Stakeholders may be categorized as [Cotterell, M. and Hughes, B]:

- Internal to the project team
- External to the project team, but internal to the organization
- External to both the project team and the organization

Other categorization divides the stakeholders into [Newman, W.M. and Lamming, M.G.]:

- Those who will use the system directly or indirectly
- Those who will be involved in developing the system

The categories above may be used to analyze the project's context and identify some stakeholders groups.

Analyzing functions (K3)

One of the techniques for stakeholders' identification is extracting all functions from the planned solution and analyzing them in the aspect of target users of those functions or individuals (or groups) who have any other stake in this.

An example may be a high-level decomposition of system's functionality and assigning owners (operators) to every single functionality.

Questionnaire (K3)

Another technique is based on questions. The following question can help in stakeholders' identification:

- Who pays for the solution?
- Who is responsible for delivering the solution within specified timeframe and budget?
- Who is responsible for the quality of the solution?
- Who will design the solution?
- Who will develop and test the solution?
- Who will maintain the solution?
- Who will use the solution?
- Who will benefit from the use of the solution?
- Who can lose because of implementing the solution?

- Who wishes to benefit but is unable to do so?
- Who impacts on the solution (today and in the future), whether positively or negatively?
- Who has rights and responsibilities over the use of the solution?
- Who would be affected by a change in the status or outputs of management?
- Who makes decisions that affect the use and status of the solution, and who does not?

Baseline and satellite stakeholders (K3)

Another common technique is based on the concept of Baseline and Satellite stakeholders. First group are Baseline stakeholders that include:

- Supplier stakeholders providing information or supporting tasks to the Baseline stakeholders
- Client stakeholders who process or inspect the products of the Baseline stakeholders

The second group of stakeholders are Satellites. They interact with the Baseline stakeholders in many ways: by communicating, applying rules or guidelines, searching for information.

A variant of the Baseline and Satellite Stakeholders approach is focused on interactions between stakeholders, not between the system and the stakeholder, as it is easier to follow.

Baseline stakeholders can be categorized into four groups (K3):

- Users, also divided into some groups: primary, secondary and tertiary users. Primary users will use the system frequently and directly, secondary users are occasional users or will use the system through an intermediary, tertiary users are those affected by the introduction of the system.
- Developers that have stake in the final requirements specification or in the system itself.
- Legislators - professional bodies, government agencies, legal representatives, quality assurance auditors.
- Decision-makers managers of the development team, user managers and financial controllers on both vendor and customer sides.

Around each of the above baseline groups there are Satellite stakeholders.

Procedure (K3)

Procedure of identification Satellite stakeholders from a Baseline [H. Sharp, A. Finkelstein & G. Galal]

1. Identify all specific roles within the Baseline stakeholder group
2. Identify supplier stakeholders for each Baseline role
3. Identify client stakeholders for each Baseline role
4. Identify Satellite stakeholders for each Baseline role
5. Repeat steps 1 to 4 for each of the stakeholder groups identified in steps 2 to 4

4.3 Techniques for Identifying Requirements (K3)	180 minutes

Terms

Apprenticing, Brainstorming, Customer's Representative, Field Observation, Interview, Questionnaire, Reuse, Self-recording, Workshop, Vision

Background

The aim of the identification of requirements is to:

- Identify all desired functions, characteristics, limitations and expectations
- Orient the requirements toward the project vision which describes the stakeholders' view of the product to be developed, expressed by the stakeholders' key needs and features.

Functions must be described clearly and unambiguously.

The functions that the customer does not want should be excluded.

Techniques for identifying requirements (K3)

Questionnaires

Questionnaires are quite easy and cheap technique, allowing to quickly obtain requirements from many users. Questionnaires do not require much effort from the questioner (as in case of verbal or telephone surveys), and often have standardized answers that allow to compile data simply and quickly.

Questionnaire consists of a several questions to be answered by the respondent. There can be open-ended or closed-ended questions. An open-ended question requires from the respondent to formulate his own answer. In case of a closed-ended question the respondent is asked to choose an answer from a number of possible options. These options should be mutually exclusive.

Types of response scales for closed-ended questions are the following:

- Dichotomous (two options)
- Nominal-polytomous (more than two unordered options)
- Ordinal-polytomous (more than two ordered options)
- (Bounded) Continuous (continuous scale)

In case of open-ended question, the answer is coded into a response scale afterwards.

For example: A question where the respondent is asked to complete a sentence.

The order of questions is important. It is not enough to put a set of questions into a questionnaire. Questions should flow logically one by one: from those more general to those more specific.

Constructing the questionnaire (K3)

When constructing the questionnaire the following rules should be considered:

- Use clear and easily understandable wording
- Formulate the questions and answers in an unambiguous way to avoid misinterpretation
- Do not make assumptions about the respondent's opinions and personality
- Ensure one question is asking about only one problem
- Formulate questions where persons with different opinions will give different answers
- Ensure that answers to a specific question are clearly different

The main disadvantage of using questionnaires is a low return rate if the respondents have no motivation to provide answers. Therefore it is important to provide proper motivation – like explaining the goal of the survey and potential advantages of including users' opinions in the planned software solution.

Interview

Interview is a conversational technique where the interviewer is asking the responder to obtain information on specified topic. This technique is very interactive and allows to modify the order of asking previously prepared questions depending on the responder's answers and the situation. For example the interviewer may skip some questions and go directly to more detailed ones, if such change would be more comfortable for the responder and effective for the interview's results.

An interview may be conducted in the following form:

- Structured interview: (called also as standardized interview) is a quantitative method of obtaining information. In such interview, responders are asked using a set of exactly the same questions in the same order.
- Semi-structured interview: a method of research where new questions can be brought up during the interview as a result of what the responder says. In semi-structured interview, the person conducting the interview has a framework of topics to be explored.

- Unstructured interview: in this method questions can be changed to meet the respondent's intelligence, understanding or belief. Questions do not include predefined and limited set of answers to be chosen by the respondent – instead the interviewer is listening (and documenting) how each individual responds to the question.

Self-recording

In this technique the stakeholder (in most cases end user who will use the software) documents his activities performed to complete specific task. For example, he documents all steps, inputs and resources needed to prepare an invoice.

In addition to document the activities "as-is" the user describes also changes, desires and needs. The advantage of this technique is low time and effort for the Requirements Engineer on the software vendor side as most of the work is done by the customer and ready results are provided for further analysis. The analysis of the results may be supported by a Business Analyst.

The main disadvantage is that automated activities are neglected and therefore not documented and considered as requirements. This is because automated activities (like printing and receiving the printed copy) may be not perceived as a part of "normal" activity. Other disadvantage of this technique is that it is highly dependent from the motivation of the user performing self-recording and his/her experience.

Representatives of the customer on site

Having representatives of the customer on site is one of the main rules in Agile methods, where close cooperation and direct communication with the customer is a key factor. This approach is one of the most effective Requirements Identification (and validation) method as it allows the representative systematically monitor the progress, verify correctness of the design and provide feedback and additional information whenever necessary.

An important thing in case of allocating the customer's representative to support vendor team is to ensure he has proper knowledge about the planned solution (about business domain, dependencies etc.) and sufficient communication skills to be able to share this knowledge and provide useful information.

Requirement's identification on the basis of existing documents

This technique can be used in case there is some already existing documentation that can help in identifying requirements within an organization. Such documentation can be:

- Process models and maps
- Process descriptions

- Organization charts
- Product specification (in the meaning of the outcome of specific process)
- Procedures (i.e. work procedures)
- Standards and instructions

Information included in such documents may be helpful in identifying requirements for the new system. For example: organizational structure may be used to establish roles and initial process owners in the new software.

The procedure of Requirements Identification on the basis of existing documents is:

1. Collect all valid documents related to processes, activities, roles, products, responsibilities, standards within an organization
2. Read the documents making initial notes and proposals for new requirements
3. Check the identified requirements together with the customer
4. Note all requirements approved by the customer

The requirements identified are base for further requirements analysis and needs to be detailed and extended with other, related and linked requirements.

This technique cannot be applied when there is no or only basic documents within an organization. It is also not recommended when the documentation is not kept up to date as it may result in creating incorrect, not needed requirements.

Reuse (Reusing the specification of a certain project)

Reusing the specification of a certain project can be done when an organization already completed one or more projects similar to the current project. Requirements specification prepared for previous project(s) can be used in another project to shorten the duration of requirements analysis and documentation and therefore – allow to start implementation earlier.

In most of the cases only some parts of existing specifications can be used in new project – as it is not exactly the same project. In any case, the documentation to be reused should be checked against the compliance with the current needs and requirements and properly adjusted.

Brainstorming

Brainstorming is a commonly used technique to obtain requirements related to not well known or new areas of an organization's activity or planned system functionality. It allows to collect many ideas from various stakeholders in short time and with low cost. During the brainstorming session the participant are submitting ideas and concepts regarding given problem. These ideas are noted down and then explored in order to select those most appropriate and accepted by the whole team. In case of difficulties in reaching agreement, the team has to make a consensus, what is done by arguing and discussing the problematic topic. The discussion is guided by the moderator, a person responsible for keeping the team on track during brainstorming session and ensuring some level of discipline and cooperation between participants.

The main risk related to this technique is that it is difficult with non-motivated participants – so it is important to involve stakeholders who are aware of advantages of implementation the software and are willing to help to make it most effective and useful.

<u>Field observation</u>

Field observation allows to watch the users' activities and processes being conducted and identify system requirements on that basis. Field observation should be performed following the steps:

- Preparation (selecting the users, agreeing on the date of the observation, preparing the list of questions need to be answered and data need to be collected)
- Conducting on-site observation (watching the users working and documenting the process, tasks and results, in some cases observation is extended with interviewing users about their jobs and the ways they realize their tasks)
    - Collecting data (identifying all possible artifacts, activities and their parameters, like duration time)
    - Identifying dependencies and limitations (of process, users etc.)
    - Indentifying group relationships in order to discover process and information flows (establishing organization, hierarchy, informal and formal links/interactions among groups, reporting relationships, etc.)
    - Identifying communication patterns (establishing who talks to whom, and how often; this is very important for systems planned to support communication within an organization)
- Analyzing data (link the overall time of completing specific task, products of the process, establishing the process flow, communication flow etc.)
- Identifying the requirements
- Evaluation of Legacy requirements

Apprenticing

The purpose of apprenticing is to elicit requirements from a customer, especially in case when the processes and activities performed by customer's personnel are not easy to describe using other techniques, like interviews, or the customer has problems with articulating the requirements about the planned software.

Apprenticing is a process of learning from the customer his job. The customer, who knows the best how to do specific job, teaches the requirement engineer – like a master and a student. The "student" is sitting down and watching the "master" working, asking questions in case of problems and explaining the tasks being performed. It is very useful when the customer is not able to provide full time support of his employees and only limited amount of their time is available for the software vendor.

The advantage of this technique is that it helps overcome the difficulty that customer's employees may have in thinking about things abstractly and expressing their tasks in words (as done in interviews) as they can always refer to the actual case and explain things on examples.

Workshops after each iteration

Workshops usually involve stakeholders representing different areas or/and domains. The goal of workshops is to review results of specified process and resolve issues that might have appeared during realization of the process. For example, the goal of workshop after designing the initial model of functionality is to check the correctness and other specific aspects of the design and discuss most important (or all if there is enough time and all necessary human resources are in place) issues and problems discovered during designing.

The most important advantage of this technique is the involvement of people who have different points of view on a given problem and therefore the most suitable solution can be found. It allows also to determine and describe requirements coming from various perspectives (for example, workshop to identify requirements involving business representatives, developers and architects can not only determine business requirements, but technical and architectural as well). In addition, identifying requirement using workshops allows to quickly discover and resolve potential conflicts between stakeholders' requirements.

The main disadvantage is that workshops require having the team sitting in one place; therefore it is difficult in case of geographically distributed teams. There may be also problems in availability of all people required to attend the workshop (i.e. another commitments, holidays etc.).

4.4 Functional and Non-functional Requirements (K3)	30 minutes

Terms

Functional Requirements, ISO 9126, NFR framework, Non-functional Requirements, Softgoal

Background

Functional requirements

Functional requirements specify what the system does. They specify the functions of the system perceived by the end user.

Functional requirements should be characterized by the following quality characteristics [ISO/IEC 25000] (K3):

- Suitability
- Accuracy
- Interoperability
- Functionality
- Compliance
- Security

Non-functional requirements

Non-functional requirements specify how the system does; they describe the quality attributes of the whole system or its specific component or function. They limit the solution i.e. by requiring specific efficiency parameters.

Non-functional requirements are difficult to describe therefore they are often vaguely expressed or not documented at all.

For example, if the requirements engineer has no knowledge and experience in describing such requirements like efficiency he/she may not be able to document them properly or may not consider them at all.

Due to the problems with expressing non-functional requirements, they may be difficult to test. Non-functional requirements should be therefore expressed clearly and be measurable.

Non-functional requirements are [ISO/IEC 25000] (K3):

- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Non-functional requirements 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 software. Functional requirements have to provide functions; non-functional requirements determine how easy and effective the functions can be used.

Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements".

Categories of non-functional requirements (K2)

There are two main categories of non-functional requirements:

- Execution qualities – can be observed at run time (like security and usability)
- Evolution qualities – embedded in the static structure of the software (like testability, maintainability, extensibility and scalability)

NFR framework (K2)

NFR (Non-Functional Requirements) is a framework for a goal modeling using the concept of softgoals requested by the customer.

Softgoals can be defined as a customer's objectives missing strict and clear criteria for satisfaction. As softgoals are usually a representation of non-functional requirements or relations between those requirements, they are defined more loosely.

Softgoals could be usability, performance, security and flexibility in a given system.

NFR framework includes an analysis of the softgoals:

- Decomposition – the softgoals are decomposed and refined to uncover a tree structure of goals and subgoals
- Discovering interfaces – once uncovering tree structures, interfering softgoals in different trees should be identified (for example: security goals may interfere with usability)

- Creating softgoal graph structure – the identified softgoal trees are formed into a softgoal graph structure
- Refinement – the final step is to pick some particular leaf softgoals in order to satisfy all the root softgoals

4.5 Description of Requirements (K3) | 30 minutes

Terms

Procedure of a Requirement's Constructing, Requirement Document

Background

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 common understanding and the best practices of the specification where templates limit the language that can be used.

Procedure of a Requirement's Constructing (K3)

Construction of a requirement is performed in the following steps:

1. Identification of the process
2. Categorization of the system activity
3. Determination of legal obligingness
4. Process refinement
5. Definition of logical, time and resource condition

Requirement document (K2)

Standard content of the requirement document is:

- Introduction
- Secrecy clause
- Standards and references
- Stakeholders
- Product objective
- Description of the system
- Functional requirements

- Non-functional requirements
- Limitations
- Interfaces
- Risks
- Acceptance

When describing requirements the following aspects should be ensured (K2):

- No redundant information
- No sophisticated and flowery phrases
- No acronyms and abbreviations that may not be known to an audience
- No colloquial expressions and shorthands
- Clear and measurable input and output definition

Typical weaknesses of requirements documents are as follows (K2):

- Trivialness
- Information out of scope
- Describing the solution (implementation details)
- Pinpointing details that unnecessary complicate the description
- No rationale (no description what has to be achieved by implementing a requirement)
- Scope and Aims are unclear

4.6 Factors affecting Requirements Engineering (K2) — 40 minutes

Background

There are many factors affecting Requirements Engineering. Some of them are related to the stakeholders on customer's side (business representatives, end-users etc.), other result from the vendor's side.

Some of most common problems influencing Requirements Engineering and coming from the customer's side are [McConnell] (K2):

- Stakeholders do not know what they want or don't have a clear idea of their requirements
- Stakeholders request new requirements after the cost and schedule have been established and approved by project management
- Communication with stakeholders is slow and not effective
- Stakeholders do not participate in reviews or are not effective on doing so
- Stakeholders are technically unsophisticated
- Stakeholders do not understand the development process
- Stakeholders do not have knowledge about present technology, its limitations and opportunities

Below is more detailed description of other factors affecting Requirements Engineering.

4.6.1. Application domain (K2)

Application domain is an important factor that may affect Requirements Engineering. Depending on the domain, different approaches may be used to collect, analyze and document requirements. Also, different levels of skills and experience may be required from the Requirements Professional. For example, building banking system would require some knowledge about banking solutions and business rules applying to this domain. If the person responsible for Requirements Identification is not familiar with the business domain, this person may not be able to obtain all important requirements, as some of them may be considered by the customer as obvious and will not be directly articulated.

Application domain may also drive the technology selection. Some kinds of solution require specific implementation techniques and technology. The person responsible for Requirements Identification, analyzing and specification should be aware of such issues and include them into the solution design. Some of such technical restrictions or requirements may significantly affect the functional requirements.

The business domain may force applying some specific standards or regulations. For instance, safety-critical systems require full requirements coverage and traceability. Some systems require the application of quality standards – like RTCA DO-178B/ED 12B "Software Considerations in Airborne Systems and Equipment Certification" for software used in civilian aircrafts. It must be also considered when identifying and analyzing requirements.

The need for flexibility has also to be stressed by the different characters of the system and the focus you need to have on creating requirements for them. A system might have more than one focus as it contains many different parts which can be handled in a different way:

- Safety Critical (risk)
- Transaction oriented (event)
- Online – real-time system (time or cycle time)
- User interaction (GUI)
- Embedded (repetitive)
- Numeric, algorithm controlled (mathematic)
- Communications and state controlled (condition)
- Administrative (flow)
- Database oriented (data)

4.6.2. Organization and Culture (K2)

Other factors affecting the Requirements Engineering process are related to the Organization and Culture. For example, the organizational structure is important when establishing permission scheme for the different users of the system or its particular functionalities.

The processes realized within an organization are also an important input to the Requirement Engineering process. The business processes will determine the basic functionality of the planned system: required functions and services, products, process owners etc. IT systems are best built when they support processes and do not follow organizations (which often change).

Culture – the way of working and communicating – is another important factor. There are different cultures and styles of working, communicating, cooperating on the world. To ensure the Requirements Engineering process is effective, the vendor company should know the customer's culture and their habits. It will allow to avoid some negative situations and conflicts and makes the work easier.

4.6.3. Organizational and Technical Maturity (K2)

The level of organizational and technical maturity also affects the Requirement Engineering process. An organization characterized by high level of maturity will usually require formal and structured approach to the Requirements Engineering, compliance with specific standards and norms, while organizations on low level of maturity would not request such requirements.

For example: standards and practices already applied in an organization will have significant impact on the Requirements Identification process, as using standard would result in some special requirements for the new software system.

4.6.4. Combination Hardware, Software and Services (K2)

In many cases there is already some software operating within the organization. When planning new software that has to be integrated with the existing IT infrastructure thi s factor must be taken into account. It may affect not only the integration interfaces and the data requirements, but other areas like efficiency requirements.

The IT infrastructure may include not only software, but also an additional hardware (example: medical equipment, banking devices). If the new system has to work with the additional hardware, the Requirement Engineer has to analyze the hardware's requirements (like interface or/and data requirements) and include them in the new system's requirements specification.

Similarly, if there are services already working within the organization, they should be identified and analyzed to check their possible connections and dependencies with the new system. The technology used in the existing services is also important, as affects the integration requirements.

In particular, the Requirement Engineer should consider the following issues:

- What interfaces between the new system and already working software, hardware or services would be the most appropriate?
- What data is to be exchanged with the other software/hardware and what format of the data would be the best?
- How can the integration affect the efficiency of the systems?

- How should exceptions be handled (i.e. a system is temporary unavailable, the external device is not available)?
- What affect on the new system would have an unavailability of other systems (like missing data, broken transactions)?
- In case of the external devices – should we ensure that the new system is compatible with devices purchased from different producers (what may result in different technology)?
- What kind of persons should work with it and in what way, Installation technicians, support persons, Trainers, Handicapped, etc. What knowledge do they have

Service Requirements

Services may be affected by the fact that they are consumed during their usage and it may be difficult for organizations to find out that in advance. Therefore service requirements are often expressed in terms of requirements regarding the delivery of the services and measurements of the services. Requirement on services is often described in a form called Service Level Agreement (SLA) which often forms part of an agreement (contract).

4.7 Differences and similarities between a general product and a specific solution (K3) — 30 minutes

Terms

Customization, “Off-the-Shelf” Software, Product

Background

General products are designed for a broad audience. Therefore they usually contain functionalities requested and used by most of potential users. The scope of functionality is usually determined by market research and feedback from the users.

Sometimes such products may be parameterized to meet the needs of specific user group. Further modifications require special patches, applications or code changes (what not always is possible).

Specific solutions are created for a concrete customer and are designed to satisfy the customer’s specific needs, expectations and requirements. Such solutions are designed for example to support the customer’s business processes, to work with specific IT infrastructure or/and technology. Custom solutions are perceived as more expensive than “off-the-shelf” software what may not be true in all situations. Such systems are usually designed for unique businesses, technology or when the customer’s needs vary from requirements of common users. Then a dedicated solution may be necessary.

Specific solutions may be developed based on a standard product, but need to be customized. Changes related for customization are usually a source of risks in the project. Dedicated solutions must be additionally tested.

<u>Similarities (K3)</u>

It is important to notice that there is a Lifecycle thinking at the creation of Requirement Engineering processes for both Product and Specific Solution and that there is a function for assigning and maintaining Requirements for different releases.

When selecting tools and methods it is important to remember, they should support the complete lifecycle and take into account the competence for different organizations who shall perform the maintenance and further development. These tools and methods should include the methods for the Requirement Analysis as well as procedures for the Requirement Management.

Most common problems (K2)

The common problem is that products are often released to production as the development of solution is completed but the necessary infrastructure is not yet developed. In this case the product would need to be upgraded when there is more customers/users operating in production and that often includes managing of different requirements collections for different customers and for different releases.

Another difficulty is to find a good process for handling requirements in product management. This may be caused by the fact, that the customer is often represented by a market function and there is a need to have both Business Case models and clear Road Maps assisting in the Requirements Analysis and Prioritizing steps.

When developing a solution there is often a need to involve a project CCB for control of Requirements and Changes. In some cases two layers of CCB are needed, one for the product (often called Product Board or Product CCB) and one for the different Release projects.

5 Specification and Documentation of Requirements (K2)	**300 minutes**

Learning Objectives for Advanced Level of Requirements Engineering
The objectives identify what you will be able to do following the completion of each module.

5.1 Documentation of Requirements (K2)

LO-5.1.1	Explain what the purpose of documenting requirements is (K2)
LO-5.1.2	Describe how requirements can be documented (K2)
LO-5.1.3	Describe the process of documenting requirements and its main activities (K2)

5.2 Specification (K2)

LO-5.2.1	Explain what a Requirement Specification is (K2)
LO-5.2.2	Describe the content of a IEEE 830 Requirement Specification (K2)
LO-5.2.3	Explain what characterizes a Requirement Specification (K2)
LO-5.2.4	Describe the process of building a Requirement Specification (K2)
LO-5.2.5	Recall standards that are important for Requirement Specification and Solution Specifications (K1)

5.3 Procedure (K2)

LO-5.3.1	Describe typical procedure of building a specification of requirements (K2)
LO-5.3.2	Build the conspectus of the Requirements Specification for a given scenario (K3)

5.4 Formalization (K2)

LO-5.4.1	Describe with examples different degrees of formalization in the specification of requirements (K3)
LO-5.4.2	Explain factors influencing the choice of degree of formalization for the specification of requirements (K2)

5.5 Quality of Requirements (K2)

LO-5.5.1	Explain with examples the consequences of errors appearing in the requirements (K2)
LO-5.5.2	Describe the common possibilities for avoiding requirement errors (K2)
LO-5.5.3	Describe the common techniques for Requirements Quality Control (K2)
LO-5.5.4	Explain the purpose and main outcomes of the Requirements Specification review (K2)

5.1 Documentation of Requirements (K2) — 40 minutes

Background

Requirements are the base for further system development and testing. They have to express customer's needs and expectations in the way that will ensure the customer's acceptance criteria are to be met. Requirements create commitment between all stakeholders regarding the scope of further work. Therefore it is very important to document requirements correctly, including all necessary information. This way both sides ensure that they have common understanding about the scope and boundary of the software product. This will allow to avoid unnecessary conflicts and misunderstandings .,

One of the major problems with Requirement Engineering is that we use the words Requirement and Requirement Specification for so many different things and on very different levels. Normally a Requirement process should be considered as an iterative process, which at some points delivers a result called Requirement Specification. Such product aims for some different things and with different scope, which implies that the requirements can be described on different levels.

Different phases for a Requirement Specification can be:

1. A Requirement specification coming out in early phases (pre-studies, prototypes, roadmaps, early Backlogs, etc) often aims to determine "is this what we want to have". It can be produced in different ways and is often compiled by a Business Analyst on a high level.
2. Then the Requirement Specification and requirements are completed in a way, that can provide a basis for internal negotiation and prioritizing.
3. The next phase focuses on making the Requirement Specification as complete as possible for the specific aim at that level (Procurements, estimation, budget, Project or Sprint planning etc). At this moment the Requirement Specification might contain a lot of process requirements: not only for the processes that shall be supported by the solution but also requirements on the supplier, requirements regarding the way of cooperation, the negotiation and prioritization between two parties. The result of this work is called Requirement Specification (the same as the input).
4. The next version of Requirement Specification is often a variant which is negotiated, prioritized, estimated and in some cases contracted. This version may be something which is a stripped version of the output from phase 3. In this step the work will concentrate on all requirements necessary to create the right product or solution (often called Requirement Specification even if it some cases has been transferred to a functional specification or description, use cases, user stories or similarly). The requirement is expressed in a more development friendly way and sometimes also assigned to some kind of solution architecture.

5. The last version is that one using for assuring the correctness of the solution and is a base for the verification and validation. It is important to notice, that it is expressed in a traceable way, often using unique numbering..

Requirement document (K3)

The minimum amount of information for each requirement should include:

- Unique reference – a unique numerical or textual identifier. This should not be changed in case the requirement is transferred, modified or deleted.
- Verification criteria – description of the test that would demonstrate to the stakeholders (especially to the customer) that the requirement has been met.
- Author – person that may be asked for clarification (or a change) if the requirement is found to be ambiguous or not clear
- Complexity – estimation of how difficult the requirement will be to implement.
- Owner – description of the individual or group that requested or need the requirement or will be the business owner of the requirement after the project is released to the production environment.
- Priority –allows to determine which requirements should be implemented first.
- Stability – determines how mature the requirement is and allows to specify whether the requirement is ready and complete enough to start work on (an example is draft version of the requirement, not enough stable to start implementation and requiring further verification with the stakeholders, further analysis and documentation).
- Status – the status of the requirement may be “proposed”, “accepted”, “verified” (with the users or other stakeholders), or “implemented”.

Additional information may include the following:

- Cost
- Assigned-to
- Revision number
- Urgency
- Reference and dependencies

Plain text may be supported with graphic elements, like initial prototypes of user interfaces and models expressing relationships between particular requirements and/or other elements of the whole system (components, modules, other systems, databases, hardware). Including graphics or models into requirements documentation allows to present the planned solution scope and its position in existing IT infrastructure within the customer's organization in more readable and unambiguous way.

Requirements should be – in every case – approved by the customer. This may require one or more review iterations, when the requirements are read and checked by both sides, possible issues discussed and resolved and all missing information provided and included in updated requirements documentation version.

Documentation of the requirements may be supported by various tools, including text processing tools (like text editor), repository and Requirements Management tools. Advanced tools allow to create detailed requirement documentation and link documents to demonstrate their dependencies. A good practice when documenting the requirements is to visualize their relationship and the context by a diagram, like UML diagram. Graphical representation is more clear and unambiguous.

5.2 Specification (K2)	60 minutes

Terms

IEEE 830, Requirements Specification, Solution Specification

Background

A specification is an explicit set of requirements to be satisfied by a material, product, or service [ASTM International].

The specification serves to track and manage requirements. In the specification, requirements are specified in a structured way and are modeled separately. Specification is a formal agreement on requirements to be implemented in the planned software system (or in other form of solution).

The term "specification" may be used in the context of requirements and solution.

There are several types of a specification. Below there is a Software Requirements Specification described.

Software Requirements Specifications (K2)

Software Requirements Specification (SRS) is a complete description of the behavior of a software system. Requirements Specification includes a set of use cases (functional requirements) describing all interactions that the users will have with the product. It also contains a supplementary and non-functional requirements.

The creation of SRS should be the customer's task. However in some cases, the vendor can support preparing the requirements specifications. This happens when i.e. the customer's organization has no qualified resources to complete this task or has no knowledge about how to create requirements specification. In such situations, the Requirements Engineer (usually together with Business Analyst) from the vendor's organization will cooperate with customer in order to produce complete and correct specifications.

General Outline of an SRS (K2) [IEEE 830]

1. Table of contents

1.1 Introduction

1.1.1. Purpose

1.1.2. Scope

1.1.3. Definitions, acronyms, and abbreviations

1.1.4. References

1.1.5. Overview

1.2. Overall description

1.2.1. Product perspective

1.2.2. Product functions

1.2.3. User characteristics

1.2.4. Constraints

1.2.5. Assumptions and dependencies

1.3. Specific requirements

1.3.1. External interfaces

1.3.1.1. User interfaces

1.3.1.2. Hardware interfaces

1.3.1.3. Software interfaces

1.3.1.4. Communication interfaces

1.3.2. Functional requirements

1.3.3. Performance requirements

1.3.4. Design constraints

1.3.5. Reliability

1.3.6. Availability

1.3.7. Security

1.3.8. Portability

1.4. Appendixes

Solution Specifications (K2)

The solution specifications are also called functional specifications.

A functional specification is a document that clearly describes the technical requirements for solution. Functional specification is the base for further system development therefore it must provide precise information about the all functional aspects of the software to be implemented. Based on it, architects and developers are able to efficiently design the technical aspects of the system. Functional specification provides guidance for testers for verification of each functional requirement (functional specification is one of test oracles).

Functional specification does not describe how the system function will be implemented and what technology to be used. It focuses on functionality, on interactions between the user and software. It is often written in a form of use case – for example: "The user enters valid input data into input fields. The user clicks the OK button. The system saves entered data and closes the dialog window". This example shows, that the specification does not define how the system saves provided data (whether calls an integration service to transfer data to other system, or saves data directly into database); it describes the interaction between the system and the user. In some cases, functional specification is delivered with description of application's screens together with "mock ups". This allows to visualize the planned design of user interface and supports implementation. Functional specification may be informal, semi-formal or formal. Detailed description of this is provided in chapter Formalization (K2).

The purpose of functional specification may be:

- Providing the base for common understanding of the scope and functionality of planned solution
- Ensuring team consensus on what the system has to achieve before starting writing source code and testing
- Providing development team with detailed description of the necessary functionality in terms of interactions between users and software
- Providing test oracle for QA team

Functional specifications are typically written after or as a part of Requirements Analysis phase.

Before starting with development work, functional specification must be approved. In order to do so, project team has to reach the consensus on the specification – meaning all issues and problems should be clarified and resolved and all team members should agree that the document is complete, accurate and understandable . The specification is declared as "complete" or "signed off". Then the development team may start writing the code and testing team prepare test cases using the functional specification as the reference.

During test execution the behavior of the system is compared against the expected behavior as defined in the functional specification.

Important standards (K1):

IEEE 1362 (System Performance Specifications), IEEE 830 (Software Requirements Specification) - IEEE 1233 (System Functional Specifications)

5.3 Procedure (K2) | 60 minutes

Background

Requirements, once identified, analyzed and modeled should be documented in clear, unambiguous manner.

A procedure of Requirements Specification includes the following activities:

1. Identification of stakeholders
2. Definition of the vision and objective
3. Requirements determination
4. Structured requirements specification
5. Description of the system environment
6. Determination of the solution area
7. Requirements analysis
8. Examination of requirements
9. Modeling of the problem area
10. Modeling of the solution area

Procedure formalizes the results of the Requirements Analysis process. The solution model is a basis for design and implementation.

The procedure involves a number of stakeholders who are supporting the specification work in different areas. For example, the customer will support identification of stakeholders and definition of the vision. The architect will contribute to description of the solution area.

The output of the procedure, the Requirements 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.

5.4 Formalization (K2) | 50 minutes

Terms

Formal Level, Non-formal Level, Semi-formal Level

Background

Requirements specification may be created on different degrees of formalization:

- Non-formal
- Semi-formal
- Formal

Non-formal approach to writing 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 and 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 base for implementation and testing as is not clear and precise enough.

Semi-formal specification will include some formal notation and will be well structured. Usually such specifications are based on specific template (often derived from relevant standards). This ensures better quality of the documents. Semi-formal specifications may express requirements in a form of models and use formalized common language. Such approach allows to express requirements in more unambiguous and precise way but due to the different nature of the employed diagrams and descriptions it is often difficult to support a comprehensive view of all functional and dynamic properties.
One of most common notations used for semi-formal documentation are UML and SysML.

Formal specification is a mathematical description of software that may be used to develop an implementation. Formal specification describes what the system should do, usually does not describe how the system should do it. As it is based on mathematical formulas and is more difficult to learn, formal specification is used quite rarely and requires mathematical knowledge.

Some of common formal specification languages are (K1):

- B-Method or Z notation
- Specification Language(VDM-SL) of the Vienna Development Method
- Abstract Machine Notation (AMN)

The degree of formalization to be applied for specific documentation depends on many factors, including:

- Organization's culture
- Internal organization's standards and rules
- Knowledge of the audience of requirements specification
- The purpose of the specification
- Necessity of adopting external regulations and norms

5.5 Quality of Requirements (K2) | 60 minutes

Terms

ISO/IEC 25000, Checklist, Quality Characteristic, Review, Validation, Verification, Traceability

Background

One of most common reasons why projects fail to deliver products on time and within assumed budget are problems with the quality of requirements. As the requirements are a base for system development, any defects or gaps propagate on the other development processes in the project. It is important to notice, that defects resulting from low quality requirements are more expensive to fix and more risky than other types of defects. In addition, the later errors are detected, the higher are the costs.

Therefore the use of verification (are we producing the product correctly) and validation (are we producing the right product) of the requirements is necessary.

Requirements should be documented and then tested against these characteristics.

Apart from the above quality characteristics there is a set of quality attributes applying to requirements. High quality requirement should be:

- Capable of being distributed
- Complete
- Consistent
- Correct
- Without a predetermined solution
- Feasible
- Measurable
- Necessary
- Prioritized
- Testable
- Traceable
- Unambiguous
- Understandable

The quality of a Requirements Specification may be improved by including the following elements:

- Outlining the purpose of the document, scope, definitions and glossary
- Outlining objectives at different levels
- Defining design and implementation constraints
- Grading/prioritization of requirements
- Clear statements of what a system should do rather than how it should do it
- Documenting legislation, assumptions and dependencies
- Avoiding supplementary descriptions of diagrams that are clear and can stand alone
- Clearly specified user catalogue and permission scheme
- Using structured presentation (readers able to find information easily)
- Using simple, clear, precise and unambiguous language

Validation (K1)

Validation is the process of confirmation that the specification of a phase or the entire system fulfils the customer's requirements. This is done to ensure that the identified requirements are compliant with customer’s needs and expectations and there is no misunderstanding between the vendor and the customer regarding the desired solution.

Verification (K1)

Verification is the process of comparison of an intermediate product with its specifications. It is thereby determined if the software was developed correctly and if the specifications that were determined during the previous phase were fulfilled. In the context of requirements, verification will include checking the format and content of requirement specification with specified standards, norms and best practices together with consistency and correctness checks.

There are many tools and techniques for requirements Quality Control. Some of them are listed below (K2).

- Prototyping. It is useful especially in case of imprecise or difficult to articulate requirements.
- Checklists. One of the most common techniques for quality control. They may include a standard set of quality elements or customized (adjusted to the specific project) quality elements that will be used to validate the requirements.

- Reviews. Other common techniques for requirements quality assurance. The Requirements Engineer should ensure that outcomes of his work have been reviewed and major issues identified and resolved as post-review follow up activity. Review is defined as an evaluation of a product or project status to ascertain discrepancies from planned results and to recommend improvements. Examples include management review, informal review, technical review, inspection, and walkthrough. [After IEEE 1028] There are several types of reviews, each may be used to verify the Requirements Engineering outcomes:
    - Peer review – a review of a work product by colleagues of the author of the product for the purpose of identifying defects and improvements
    - Technical review – group discussion focused on achieving consensus on the technical approach to be taken
    - Walkthrough – step-by-step presentation by the author of a document in order to gather information and to establish a common understanding of its content [Freedman and Weinberg, IEEE 1028]
    - Inspection – type of formal review that relies on visual examination of documents to detect defects, e.g. Violations of development standards and non-conformance to higher level documentation
    - Demonstrations – (so called Demos) they are a review technique often used in Agile approaches and Scrum to get customer and other stakeholders view on an intermediate result and to collect changes needed for the final solution. In early stages a subject of demo can be just paper or PowerPoint presentation.
- Traceability. This is another technique supporting the quality of requirements. Traceability allows to verify if all requirements have associated test cases to cover them during test execution. If not – this may mean that requirements are not testable and should be corrected to enable testing their implementation.

6 Requirements Analysis (K3) — 500 minutes

Learning Objectives for Advanced Level of Requirements Engineering

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

6.1 Object-Oriented Analysis and Design (K3)

LO-6.1.1	Explain the purpose of Object-Oriented Analysis and Design (K2)
LO-6.1.2	Explain and provide examples of when Object-Oriented Analysis and Design can help Requirements Analysis better than other approaches (K2)
LO-6.1.3	Explain the main characteristics of the UML notation (K3)
LO-6.1.4	Understand the purpose and application of different UML diagrams (K3)
LO-6.1.5	Explain the main characteristics of the SysML notation (K3)
LO-6.1.6	Analyze basic UML models for a given scenario (K4)
LO-6.1.7	Analyze basic SysML models for a given scenario (K4)

6.2 Cost Estimates (K3)

LO-6.2.1	Explain the reasons of the cost estimation (K2)
LO-6.2.2	Describe the procedure of cost estimation using different method (K3)
LO-6.2.3	Conduct cost estimation using the Use Case Points method (K3)
LO-6.2.4	Explain and provide examples what factors can influence cost estimates (K2)

6.3 Prioritization (K2)

LO-6.3.1	Explain the common methods of Requirements Prioritization (K2)
LO-6.3.2	Describe with examples the procedure for prioritizing (K2)
LO-6.3.3	Conduct requirements prioritization for given scenario (K3)

6.4 Agreeing on Requirements (K2)

LO-6.4.1	Describe what should be considered when agreeing on requirements (K2)
LO-6.4.2	Explain the reasons of agreeing on requirements (K2)
LO-6.4.3	Describe the process of agreeing on requirements (K2)

6.5 Conflict Management (K3)

LO-6.5.1	Explain why a Conflict Management is necessary (K2)
LO-6.5.2	Explain the Conflict Management Model (K2)
LO-6.5.3	Give examples of different techniques used to manage conflicts (K3)
LO-6.5.4	Describe the phases of a Conflict Management (K3)

6.6 Conflict Identification (K2)

LO-6.6.1	Explain how can conflict be identified (K2)
LO-6.6.2	Recall different types of conflicts (K1)

6.7 Conflict Analysis (K3)

LO-6.7.1	Explain the purpose of a Conflict Analysis (K2)
LO-6.7.2	Describe common techniques used to conduct a Conflict Analysis (K3)
LO-6.7.3	Describe the main outcomes of a Conflict Analysis and their role in decision making (K3)

6.8 Conflict Resolution (K3)

LO-6.8.1	Explain the purpose and meaning of a Conflict Resolution (K2)
LO-6.8.2	Apply the common strategies of a Conflict Resolution (K3)

6.9 Techniques and Strategies for Conflict resolution (K2)

LO-6.9.1	Describe common techniques and strategies for a Conflict Resolution (K2)
LO-6.9.2	Describe the main factors to be considered when selecting a technique for a Conflict Resolution (K2)

6.1 Object-Oriented Analysis and Design (K3)	120 minutes

Terms

Activity Diagram, Analysis Methods, Analysis Models, Behavioral Diagram, Class Diagram, Communication Diagram, Component Diagram, Composite Structure Diagram, Deployment Diagram, Interaction Overview Diagram, Object Diagram, Object-Oriented Analysis, Object-Oriented Design, Package Diagram, Parametric Diagram, Requirements Diagram, Sequence Diagram, System Analysis, State Machine Diagram, Structural Diagram, SysML, Timing Diagram, UML, Use Case Diagram

6.1.1 Introduction (K2)

Object-Oriented Analysis and Design (OOAD) is a software engineering approach that models a system as a group of interacting objects. Each object represents some entity in the system being modeled and is described by its class, state (data elements) and behavior.
OOAD provides various models allowing to show, for example, the static structure or dynamic behavior of the collaborating objects.

Object-Oriented Analysis and Design comprises with two sub-disciplines:

- Object-Oriented Analysis (OOA) focusing on analyzing the functional requirements for a system
- Object-Oriented Design (OOD) developing the analysis models to deliver detailed implementation specifications

System Analysis (K2)

System Analysis describes a system and its limitations to the environment. It provides a well-founded understanding of the environment and the system requirements. System Analysis deals with the following characteristics:

- Functions
- Processes
- Usage scenarios
- Interactions
- Division into sub-systems and modules
- Interfaces (i.e. with external systems)
- Data

- Working documents
- Rules (like business rules)
- Perspectives and points of view
- Requirement Model
- Solution Model

Analysis methods and models (K2)
When choosing an analysis model or a combination of models, it is important to consider the application domain or the target operating environment as well as other factors in described in Factors affecting Requirements Engineering (K2).

Analysis method	**Analysis model**
Context analysis	Context model
Architectural analysis	Functional decomposition
Data stream analysis	Data stream model
Condition analysis	Condition model Petri net
Decision analysis	Decision table Petri net
Data analysis	Semantic data model Entity-relationship model Data dictionary
Object-oriented analysis	Use case diagram Activity diagram Sequence diagram Class diagram

6.1.2 Object-oriented Analysis (K3)

UML (K1)

Unified Modeling Language is a unified notation for the analysis and design of systems. It contains different diagram types for different views on the system (structural or behavioral diagrams).

Behavioral diagrams (K3)

Behavioral diagrams depict behavioral features of a system or business process. These diagrams include the following diagram types:

- Activity Diagrams. Activity diagrams model the behaviors of a system, and the way in which these behaviors are related in an overall flow of the system.
- Use Case Diagrams. Use Case diagrams capture Use Cases and relationships among Actors and the system; they describe the functional requirements of the system, the manner in which external operators interact at the system boundary, and the response of the system.
- State Machine Diagrams. State Machine diagrams illustrate how an element can move between states, classifying its behavior according to transition triggers and constraining guards.
- Timing Diagrams. Timing diagrams define the behavior of different objects within a time-scale, providing a visual representation of objects changing state and interacting over time.
- Sequence Diagrams. Sequence diagrams are structured representations of behavior as a series of sequential steps over time. They are used to depict work flow, message passing and how elements in general cooperate over time to achieve a result.
- Communication Diagrams. Communication diagrams show the interactions between elements at run-time, visualizing inter-object relationships.
- Interaction Overview Diagrams. Interaction Overview diagrams help to visualize the cooperation between other interaction diagrams to illustrate a control flow serving an encompassing purpose.

Structural diagrams (K3)

Structural diagrams depict the structural elements composing a system or function. These diagrams reflect the static relationships of a structure, such as Class or Package diagrams, or run-time architectures such as Object or Composite Structure diagrams.

Structural diagrams include the following diagram types:

- Class Diagrams. Class diagrams capture the logical structure of the system, the Classes and objects that make up the model, describing what exists and what attributes and behavior it has.
- Composite Structure Diagrams. Composite Structure diagrams reflect the internal collaboration of Classes, Interfaces and Components (and their properties) to describe a functionality.
- Component Diagrams. Component diagrams present the pieces of software, embedded controllers and such that make up a system as well as their organization and dependencies.
- Deployment Diagrams. Deployment diagrams show how and where the system is going to be deployed; that is, its execution architecture.
- Object Diagrams. Object diagrams present object instances of Classes and their relationships at a point in time.
- Package Diagrams. Package diagrams depict the organization of model elements into packages and the dependencies amongst them.

SysML (K3)

System Modeling Language is a special modeling language for System Engineering. It is an extension of UML 2.1.

SysML offers some improvements over UML. These improvements are the following:

- More flexible and expressive semantic. SysML reduces UML's software-centric restrictions and adds two new diagram types, requirement and parametric diagrams. SysML allows to model a wide range of systems which include hardware, software, information, processes, personnel and facilities.
- Easy to learn and apply. SysML do not utilize many of UML's software-centric constructs, therefore the language is smaller than UML - both in diagram types and total constructs.
- Supporting models and views. Models and views extend UML's capabilities and are architecturally aligned with IEEE-Std-1471-2000.
- Reusing seven of UML’s diagrams and providing two new diagrams (requirements and parametric diagrams) for a total of nine diagram types.

The Requirement diagrams allow to capture functional, performance and interface requirements. The Parametric diagrams can be used to define performance and quantitative constraints.

Other Diagrams that also can be valuable for the Requirement Analysis are:

- Process Diagram
- Workflow Diagram
- Dataflow Diagram

6.2 Cost Estimates (K3) | 60 minutes

Terms

Algorithmic procedure, Analogy, CoCoMo, Delphi method, Function points, Program Evaluation and Review Technique, Putnam formula, Use Case points, Weighted Micro Function Points

Background

The cost of the project depends on various factors:

- Project type
- Process maturity
- Design methods and tools
- Technology
- Complexity of the planned solution
- Quality objectives (for example desired level of software quality)
- Team qualifications
- Team distribution
- Experiences (New Method, new Technology, new Tools)

The exactness of the cost estimations depends upon the progress of the project and its maturity. Cost estimation can be conducted using:

- Analogy
- Algorithmic procedure

Conclusion by analogy (K3)

Cost estimation is based on comparison with similar project's costs. It 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 of requirements
- Scope of the solution
- Technology used
- Personnel's characteristics (skills, experience)

Based on the results of comparison, Project Manager can make an estimate.

Algorithmic procedure (K1)

- Putnam formula
- Function Points
- Use Case Points
- Cost Constructive Model (CoCoMo)
- Delphi method
- Weighted Micro Function Points (WMFP)
- Program Evaluation and Review Technique (PERT)

Use Case Points (K3)

The Use Case Points (UCP) is an estimation method that provides the ability to estimate an application's size and effort from its use cases. UCP analyzes the use case actors, scenarios and various technical and environmental factors and abstracts them into an equation.

The Use Case Points method is based on the use case model, which consists of actors and use cases. The number and weight of the use cases identified is the most important component in the calculation of the Unadjusted Use Case Points.

The size of a software system is calculated from the Unadjusted Use Case Points by adjusting them with the technical complexity factor.

The formula for calculating the size of a system is composed of four variables:

- Technical Complexity Factor (TCF)
- Environment Complexity Factor (ECF)
- Unadjusted Use Case Points (UUCP)
- Productivity Factor (PF)

The complete equation is:

UCP = TCP * ECF * UUCP * PF

Procedure of Use Case Points (K3)

Steps to generate the estimate based on the UCP method:

1. Determine and compute the Technical Factors
2. Determine and compute the Environmental Factors
3. Compute the Unadjusted Use Case Points
4. Determine the Productivity Factor
5. Compute the product of the variables

Technical Complexity Factors (TCF)

There are thirteen standard Technical Factors. Each factor is weighted according to its relative impact (weights from 0 – indicating the factor is irrelevant; to 5 – for factors with the most impact).

Technical Factor	**Description**	**Weight**
T1	Distributed system	2
T2	Performance	1
T3	End User Efficiency	1
T4	Complex internal Processing	1
T5	Reusability	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent	1
T11	Special security features	1
T12	Provides direct access for third parties	1
T13	Special user training facilities are required	1

The technical factors are evaluated by the development team.

Environmental Complexity Factor (ECF)

Environmental Complexity factor estimates the impact on productivity that various environmental factors have on an application. Each environmental factor is evaluated and weighted according to its perceived impact and assigned a value between 0 and 5.

Environmental Factor	**Description**	**Weight**
E1	Familiarity with UML	1.5
E2	Application Experience	0.5
E3	Object Oriented Experience	1
E4	Lead analyst capability	0.5
E5	Motivation	1
E6	Stable Requirements	2
E7	Part-time workers	-1
E8	Difficult Programming language	2

Unadjusted Use Case Points (UUCP)

Unadjusted Use Case Points are computed based on:

- Unadjusted Use Case Weight (UUCW)
- Unadjusted Actor Weight (UAW)

Unadjusted Use Case Weight (UUCW)

Use Cases are divided to three categories: Simple, Average or Complex. The weight of each of category depends on the number of steps included in the use case (including alternative flows).

Use Case Type	**Description**	**Weight**
Simple	Has a simple user interface and involves only a single database entity; its success scenario has up to 3 steps; its implementation involves less than 5 classes.	5

Average	Has more interface design and involves 2 or more database entities; between 4 to 7 steps; its implementation involves 5 to 10 classes.	10
Complex	Has a complex user interface or processing and involves 3 or more database entities; more than seven steps; its implementation involves more than 10 classes.	15

Unadjusted Actor Weight (UAW)

Analogically, there are three types of the Actors: Simple, Average or Complex. This categorization is based on the interactions.

Actor Type	**Description**	**Weight**
Simple	The Actor represents another system with a defined API.	1
Average	The Actor represents another system interacting through a protocol, like TCP/IP.	2
Complex	The Actor is a person interacting via an interface.	3

Productivity Factor (PF)

The Productivity Factor is a ratio describing the number of man hours per Use Case Point. This ratio is based on past projects. If there is no historical data, it is recommended to use a number between 15 and 30. A typical value is 20.

Agile estimations

In Agile projects often managed by Scrum, there is an estimation method called Planning Game or Playing Poker. This method is used to gain consensus in the team. The team ability is then measured through something called Bum Down Rate and improved through Retrospective sessions conducted after every sprint where planned figures are compared to the actual. This allows to improve the team's ability to estimate. The productivity factor for an Agile/Scrum team is often called the Velocity.

6.3 Prioritization (K2)	30 minutes

Terms

Analytic Hierarchy Process, Cost Value Approach, Scale, Three-Level Scale

Background

High quality requirements should be explicitly prioritized. It allows to ensure that the software product contains the most essential functions, even if timelines are short and resources limited. Prioritization allows to establish requirement's relative importance and plan the implementation to complete the most crucial requirements first.

The customer and the vendor team must collaborate on requirements prioritization, as the vendor team do not always know which requirements are most important to the customers and – from the other side – the customer is not able to determine the cost and technical difficulty associated with specific requirements.

Customers prioritize requirements initially from the perspective of how much is each one valuable for them.

Prioritizing of requirements (K2)

Prioritizing of requirements is a sub process of the release planning process. The release planning process includes the following activities:

- Requirement's prioritization
- Requirements selection
- Determining release requirements
- Validating release requirements
- Preparing the release package

Roles (K1)

Prioritization process usually involves the following roles:

- Project Manager
- Key customer representatives

- Business Analysts
- Requirement Engineers
- Development representatives

Scales approach (K2)

A common approach to prioritization is to group requirements into priority categories. Usually the three-level scale is used (example: 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.

Examples of three-level scale are (K2):

Name	Description
High	a critical requirement, required for the first software release
Medium	supports necessary system operations but could wait until a later release if necessary
Low	a functional or quality enhancement, so called "would be nice to have"

Name	Description
Essential	the product is not acceptable unless these requirements are fulfilled
Conditional	would enhance the product, but the product is not unacceptable if absent
Optional	functions that may or may not be worthwhile

Cost Value Approach (K2)

One of the other methods for prioritizing requirements (especially in the implementation phase of the project) is the Cost Value Approach. The main concept of this approach is to determine for each individual candidate requirement what the cost of implementing the requirement would be and how much value the requirement has.

The assessment of requirement's values and costs is performed using the Analytic Hierarchy Process (AHP). This method involves requirement's pairwise comparison. For each of pair of (candidate) requirements there is a value or cost assessment performed, comparing the one requirement of a pair with the other.

For example, a value of 5 for (Requirement 1, Requirement 2) indicates that Requirement 1 is valued five times as high as Requirement 2.

<u>Cost Value Approach Procedure (K2)</u>

The Cost Value Approach includes five steps for reviewing requirements and determining a priority among them. These steps are the following:

1. Requirement Engineers review candidate requirements to verify the completeness and to ensure that they are described in an unambiguous way.
2. Business stakeholders (usually the customer's and/or user's representatives) asses the relative value of the candidate requirements using AHP method.
3. Software Engineers estimate the relative cost of implementing each candidate requirement using AHP method.
4. Software Engineers calculate each candidate requirement's relative value and implementation cost using AHP method and plots these on a cost-value diagram. The value is represented on the y axis of this diagram and the estimated cost on the x-axis.
5. The cost-value diagram is used as a conceptual map for analyzing and discussing the candidate requirements. Project Manager with the support of Requirement Engineers, Software Engineers and the customer prioritize the requirements and decide which will be implemented.

Other Prioritization Techniques (K1):

- Quality Function Deployment (QFD)
- Planning game (PG)
- PROMETHEE (Preference Ranking Organization METHod for Enrichment Evaluation)
- 100-point method (100P)
- Planning Game combined with AHP (PGcAHP)
- MoSCoW Method (Must have this, Should have this, Could have this, Would have this)
- Wiegers' method

6.4 Agreeing on Requirements (K2) | 30 minutes

Terms

Signoff

Background

Obtaining requirements signoff is typically the final task of Requirements Analysis and Design.

Agreeing on requirements, often called requirements signoff is a formal agreement that the content and scope of the requirements are accurate and complete. The requirements signoff should be done by project's stakeholders, including:

- Project Managers on both the customer and the vendor sides
- The customer's business representative
- The Business and System Analysts
- The Requirements Engineers
- The representatives of Quality Assurance and development teams.

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

Obtaining requirements signoff typically involves a face-to-face final review of requirements with the project's stakeholders. After the review, the stakeholders are asked to formally approve the reviewed requirements document.

If the review discovers problems in the documentation, the documentation should be corrected and the new version should be provided to the reviewers.

In most cases involving all the stakeholders in reviewing all requirements documents is neither possible nor effective. For example, if many requirements apply to only a subset of stakeholders, it may be more practical to ask each stakeholder to approve only selected set of the requirements.

Agreeing on requirements is considered complete, when all relevant project's stakeholders have signed off the requirements document. Completion of requirements signoff should be communicated to the project team and usually is a project milestone.

6.5 Conflict Management (K3) | 60 minutes

Terms

Conflict, Conflict Management, Conflict Management Model

Background

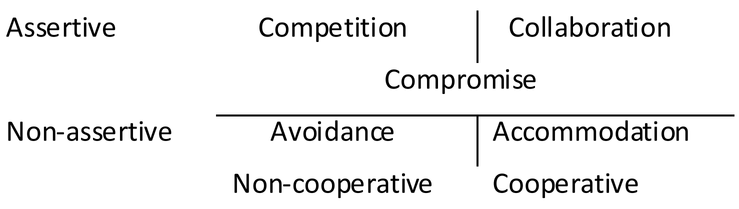
Conflict is when two or more values, perspectives or opinions are contradictory in nature and haven't been aligned or agreed about yet.

A typical conflict for most software products occurs between the usability and security requirements.

Conflict is inevitable and in some cases provides benefits. In the context of Requirements Engineering, conflict may appear between stakeholders having contradictory requirements or the proposed solution for requirements' implementation may be contradictory.

Conflict Management refers to the long-term management of conflicts. It includes the variety of ways by which people handle conflicts.

<u>Conflict Management Model (K3)</u>



6.6 Conflict Identification (K2) — 40 minutes

Terms

Adapted Planning Matrix, Timeline, Types of Conflict

Background

The first step in Conflict Management is Conflict Identification. This is caused by the fact, that only establishing the real cause, source and background of the conflict allow analyzing it and finding an effective resolution.

<u>Tools (K2)</u>

Tools for identifying potential conflicts:

- Adapted Planning Matrix – uses the hierarchy of objectives from a project-planning matrix to identify potential conflicts
- Timeline – encourages participants to consider past conflicts and to think about whether they could emerge again or whether they have been resolved in a satisfactory way

<u>Typology of conflicts (K2)</u>

Establishing the type of a conflict helps to understand its source and work on the possible solution.

There are a number of types of conflicts. In case of Requirements Engineering the most applicable types of conflicts are related to:

- Levels of interaction
    - Conflicts at the intra-community level (conflicts between members of the same community)
    - Conflicts at the inter-community level (conflicts between two communities or members of two separate communities)
    - Conflicts at the local level
    - Conflicts at the national level
    - Conflicts at the international level
- Causes and Sources of Conflict

-
  - Interest conflicts (related to the actions and emotions by which people become involved to gain or protect their needs)
  - Information conflicts (resulted from lack of information or differences in same information)
  - Relationship conflicts (caused by differences of personality and emotions, misperceptions, stereotypes and prejudices)
  - Structural conflicts (differing ideas concerning process, rules and power to control time and space)
  - Value conflicts (differences between cultural, social or personal beliefs or different world views and traditions)
- Topic-based typology of conflicts
  - Conflicts over access (consequences of a change in access or inequitable access)
  - Conflicts due to change in resource quality and availability (resulting from a change in the amount or the quality of the resource available to the different parties)
  - Conflicts regarding authority over resource (caused by shifting in decision-making authority over a resource)
  - Conflicts that result from differences in use and non-use economic values
  - Conflicts associated with information processing and availability
  - Conflicts resulting from legal/policy interests

6.7 Conflict Analysis (K3) — 60 minutes

Terms

Root Causes, Stakeholder Assessment

Background

Analyzing conflicts includes the following activities:

- Stakeholder Assessment

To analyze a conflict and develop strategies for conflict resolution, there should be a basic assessment of stakeholders interests performed. Identification of needs, expectations and interests of the different people/groups involved in a conflict or conflict-bearing situation is crucial to establish the possible causes of the conflict, as well as acceptable ways to solve it

- Root Causes identification

Factors which seem to be the cause of a conflict may not be the real root causes. Participants need to develop sensitivity to this and learn how to identify causes and root causes of conflicts.

The root cause can be discovered during the assessment of stakeholders' interests.

Developing a strategy for conflict resolution

After an assessment of stakeholders has been made and a draft analysis of the conflict has taken place, a strategy for conflict resolution can be developed. This session demonstrates the process of developing such a strategy.

It is important to create some type of ownership over the different Requirement to get the strategy right.

6.8 Conflict Resolution (K3) | 60 minutes

Terms

Arbitration, Diplomacy, Mediation, Negotiation

Background

Conflict Resolution is a range of methods used for eliminating sources of conflict. Another term to express "conflict resolution" is "dispute resolution" or "alternative dispute resolution".

Processes of Conflict Resolution include the following:

- Negotiation
- Mediation
- Diplomacy
- Arbitration

Conflict can be resolved using the following techniques [Thomas and Kilmann]:

- Accommodation - surrender one's own needs and wishes to accommodate the other party
- Avoidance – avoiding the conflict by ignoring it, changing the subject etc. This technique can be used as a temporary solution to buy time or as a mean of dealing with very minor conflicts
- Collaboration – working together to find a mutually beneficial solution
- Compromise – discuss the problem with the presence of the third party in order to find a solution acceptable for both parties
- Competition – assert one's viewpoint at the potential expense of another

In resolving conflict, it is important to ensure the following rules are applied:

- The causes of the conflict must be clearly articulated
- The reasons to work on conflict must be clearly articulated
- The way of resolving the conflict should be communicated

- The issues should be addressed face-to-face (notes, phone call or email correspondence are not a productive way to resolve conflicts)
- In some cases, the discussion about the conflict should be postponed. In the resolution of a conflict, emotions may interfere with arriving at a productive resolution. If this case it is recommended to take a time-out and resume resolving the conflict at another designated time.

When dealing with a conflict, the following negative attitudes and communication patterns should be avoided:

- Avoiding conflict altogether rather that discussing. This usually causes more stress to both parties, as tensions rise and a much bigger argument eventually occurs.
- Being defensive
- Over generalizing (using statements like “You always…” and “You never…”, “You never do what I want you to do!”)
- Bringing up past conflicts
- Forgetting to listen
- Criticizing and blaming the other person for the situation

6.9 Techniques and Strategies for Conflict Resolution (K2)

40 minutes

Background

If the conflict appears, the Requirements Manager should undertake appropriate actions in order to resolve it and get consensus. The reason of the conflict may vary. Most common causes of a conflict are the following:

- The requirements coming from different stakeholders are contradictory
- The stakeholders cannot reach consensus about the desired implementation of the requirements
- In case of requirements or change prioritizing, the stakeholders are not able to agree on the priorities
- The stakeholders request to change already agreed and approved requirements without formal Change Management process

Conflict Resolution Techniques (K2)

Some techniques for Conflict Resolution are:

- Listening and understanding the nature of conflict – it is a very important point as to resolve the conflict it must be understood. The Requirements Manager should first listen to the people involved in the conflict and understand its nature and source.
- Group meeting – one of the best techniques to resolve a conflict is arranging a meeting of all the involved stakeholders and discussing the problem. In the meeting everyone should have a chance to speak out about their views. The discussion is a good method of understanding what was a reason for the conflict and how it can be resolved. A group meeting allows also clearing off any misunderstanding between people involved in the conflict as they can listen to the other parties and compare different points of view.
- Analyzing needs and priorities – usually done during a group meeting. This technique allows to identify and analyze needs and priorities of different people involved in the conflict and agree on a solution that will be acceptable for everyone. Establishing the priorities (like requirement's implementation priorities) may be managed by a Project Manager, who will verify the capability of satisfying particular expectation from project's budget and cost point of view.

- Involving external party – in some cases involving external, independent party is necessary. Such person is not personally involved in the conflict and may help in analyzing the problem from quite new perspective.

It is also important to ensure that the resolution is clearly stated and communicated to all involved participants to assure a formal approval of the solution to the conflict.

7. Quality Control of Requirements (K4)	400 minutes

Learning Objectives for Advanced Level of Requirements Engineering
The objectives identify what you will be able to do following the completion of each module.

7.1 Quality Control Activities and Techniques (K3)

LO-7.1.1	Explain the role of Quality Control in Requirements Engineering (K2)
LO-7.1.2	Describe different Quality Control activities applicable for requirements (K2)
LO-7.1.3	Use a procedure of Quality Control for a given scenario (K3)

7.2 Effective Control of Requirements (K3)

LO-7.2.1	Explain the meaning of the control of requirements (K2)
LO-7.2.2	Describe and give examples of different techniques and tools supporting effective control of requirements (K3)

7.3 Change Management (K3)

LO-7.3.1	Describe the process of Change Management and the role of Change Control Board (K2)
LO-7.3.2	Explain on examples the procedure and main outputs of Impact Analysis (K3)
LO-7.3.3	Describe the role of Impact Analysis in decision making (K3)
LO-7.3.4	Explain on examples the procedure and main outputs of Change Initiation (K2)
LO-7.3.5	Describe the purpose and content of main documents used in Change Management (K2)
LO-7.3.6	Describe using examples change life cycle (K3)
LO-7.3.7	Explain on examples the procedure and main outputs of Change Prioritization (K2)
LO-7.3.8	Explain on examples the procedure and main outputs of Change Consolidation (K2)
LO-7.3.9	Build basic Change Management process for given scenario (K3)

7.4 Review methods (K3)

LO-7.4.1	Explain common review methods and their possible application in specific situations (K3)
LO-7.4.2	Differentiate between different review methods (K3)
LO-7.4.3	Decide what review method is best applicable for a given scenario (K3)
LO-7.4.4	Plan review activities and outcomes for a given scenario (K3)

7.5 Prototyping and simulation methods (K2)

LO-7.5.1	Recall what prototyping and simulation are and how they can be applied in a project (K2)
LO-7.5.2	Explain the goals of Prototyping and simulation methods (K2)
LO-7.5.3	Describe how prototyping and simulation methods can support Requirement Engineering and improve the solution design (K2)

7.6 Different validation techniques (K2)

LO-7.6.1	Describe the purpose of validation (K2)
LO-7.6.2	Describe different validation techniques (K2)

7.1 Quality Control Activities and Techniques (K3) — 60 minutes

Terms

Quality Control, Software Quality Assurance, Verification and Validation Requirements Matrix, Requirements Traceability Matrix

Background

<u>Quality Control (K1)</u>

Quality Control is defined as:

"Operational techniques and activities that are used to fulfill requirements for quality."

ISO 9000:2000 defines Quality Control as:

"Part of quality management focused on fulfilling of quality requirements."

The goal of a Quality Control is through use of operative methods steer and control the quality of products or services so they meet specified standards of quality. Typical such operative methods involved in Requirement Engineering are:

- Project Management
- Risk Management
- Change Management
- Verification and Validation, including reviews
- Verification and Validation Requirement Matrix and Requirements Verification Matrix
- Configuration management and Requirement tracking

In Requirements Engineering, Quality Control also focuses on verifying if the produced requirements documentation meets relevant quality criteria. Such criteria may be:

- Common requirements specification quality criteria: completeness, consistency, modifiability and traceability
- Compliance with internal or external standards
- Compliance with stated user needs
- Ability to implement (agreed with development team)

Quality Control includes the following:

- Verification and Validation
- Verification and Validation Requirements Matrix and Requirements Verification Matrix
- Software Quality Assurance (including reviews and audits)

Software Quality Assurance (K3)

Software Quality Assurance provides assurance that the software products and processes in the project life cycle conform to their specified requirements. This is done by planning and performing a set of activities together with using selected tools and techniques to deliver confidence that quality is being built into the products of software development process.

One of the main concepts of Quality Assurance is to ensure the best possible quality by realizing specified activities at each stage of the project what would allow to identify potential problems as soon as possible. The role of Quality Assurance is to ensure that processes are appropriate and correct (resulting in providing desired outputs and performed according to the process requirements) and later implemented according to plan. The level of quality is measured by relevant measurement processes, defined in Quality Assurance plan.

Verification and Validation (K3)

Software Verification and Validation is a disciplined approach to assessing software products throughout the product life cycle. Verification and Validation effort strives to ensure that quality is built into the software and that the software satisfies user requirements [IEEE 1059-93].

The process determines whether or not products of a given development or maintenance activity conform to the requirement of that activity, and whether or not the final software product fulfills its intended purpose and meets user requirements.

Verification is an attempt to ensure that the product is built correctly, in the sense that the output products of an activity meet the specifications imposed on them in previous activities.

Validation is an attempt to ensure that the right product is built, that is, the product fulfills its specific intended purpose.

Both the verification and the validation processes should begin as early as possible in the project. They should be carefully planned to ensure that each identified resource, role and responsibility is clearly assigned to defined verification and validation tasks. Planning includes defining techniques and tools to be used.

Typical Verification and Validation activities is:

- Reviews
- Checklists
- Tests
- Acceptance
- Audits
- Inspections
- Etc.

The standard content of verification and validation plans is specified in IEEE1012-98: s7 and IEEE1059-93: Appendix A [IEEE 1012 and IEEE 1059].

VVRM and RTM (K3)

Verification and Validation Requirements Matrix (VVRM) (K3)

A Verification and Validation Requirements Matrix (VVRM) provides traceability from individual requirements to the relevant testing results. Therefore VVRM creates a basis for making decisions on acceptance and ensuring the stakeholder need is met.

VVRM should include reference to:

- Requirements (both functional and non-functional)
- Evidence and information (Test case, results)
- Acceptance criteria and recommendation
- Responsibilities
- Progress and key milestones

The main purpose of VVRM is to provide a tool to manage incremental verification together with a definition of each of the incremental test stages and stage criteria designer for specific requirements.

Requirement Traceability Matrix (RTM) (K3)

Another useful matrix used in Quality Control, is Requirements Verification Matrix or Requirement Traceability Matrix.

Requirements Traceability Matrix (RTM) is a tool for ensuring that the scope of the project, its requirements and outcomes remain unchanged when compared to the baseline. RTM monitors (traces) the outcomes by establishing a thread for each requirement- from the project's initiation to the final implementation.

7.2 Effective Control of Requirements (K3) — 60 minutes

Background

Control of requirements is necessary to ensure the solution will satisfy the customer's needs. In particular it allows to ensure:

- All requirements are unique and identified
- All requirements are under Configuration Control
- All requirements are under Version Control
- All requirements are up-to-date
- No longer valid requirements are properly marked and not included in the implementation plan
- Dependencies between requirements and between requirements and other work products (like test cases) are clear and up-to-date

There are numerous techniques and tools supporting effective control of requirements:

- Traceability tools
- Configuration and Change Control tools
- Requirements Management tools

The control of requirements may be performed using the following tools:

- Reviews
- Audits
- Checklists

Traceability is an important administrative activity as well as a prerequisite for verification and validation. It is also necessary for the Change Management process when the effects of a change should be analyzed.

7.3 Change Management (K3) | 120 minutes

Terms

Change, Change Control Board, Change Analysis, Change Initiation, Change Consolidation, Change Management, Change Request

Background

The Change Management process is the process of requesting, determining attainability, planning, implementing, and evaluating of changes to a software system, documents or other products of the project. The purpose of Change Management is to allow and support the processing of changes and ensuring traceability of changes.

Changes of the requirements may be requested in any time during the realization of the project and after releasing the final software on the production environment. Changes will always happen and it is important to plan changes in the terms of the process and time. A normal run project may have about 15% of the effort allocated on managing and implementing changes. In case of an Agile changes are even much more frequent.

The source of change may be the following:

- Extension of existing functionality
- Defects found in the software/documentation
- Requested new functionality
- Change of existing functionality
- Changes resulting from external factors (organizational changes, regulatory changes)

Change Management depends on the different price models used such as Fix Price, Time and Material or Target Price.

<u>Change Management Process (K2)</u>

Change Management process includes the following activities:

- Identification of potential change
- Requesting new functionality
- Analysis of the change request

- Evaluating the change
- Planning the change
- Implementing the change
- Reviewing and closure of the change

Change Request (K2)

A change should be submitted as a formal Change Request document (referred also as Request For Change (RFC)). Such document should describe the reason for change and requested solution together with additional details like:

- The name of person/department or other entity requesting the change
- Submission date
- Planned date of implementation of the change

Changes can be a request for a new feature, but they are normally handled in the same way.

The originator of a change may be any stakeholder on both customer and vendor side: users, customer, Project Manager, Business Analyst, developers, testers, architects etc.

Change Request should be submitted to members of Change Control Board, who will analyze the request and make decision about further actions.

The role of Change Control Board is to analyze potential changes and make decisions regarding Change Requests. CCB is a group of individuals within a project group who are responsible for making the decision as to if and when any changes are to be made in regards to work products or schedule events. Large changes of the requirements can be so serious that they represent a contractual change therefore careful analysis of the potential impact of the change may be crucial.

The Change Control Board decides about changes in two steps. The first one is to analyze the impact of the proposed changes. After completing this evaluation Change Control Board either approves or rejects the changes. In some cases they may request more information before making final decision or defer the decision until some other occurrences that would affect the choice take place. Significant and complex changes that will affect baselines should be always put through the CCB for approval.

There may be one CCB for the product changes (Sometimes called Product Board) and one for the different Release projects.

Change Control Board (K1)

The Change Control Board for a project can consists of:

- Project management
- Business and system analyst
- Development
- Quality assurance
- Business management, if applicable
- Customer representative, if applicable

The Change Control Board for a product the following roles may involved in addition to the above:

- Product Manager
- Market Representatives
- Product Engineers or Requirement engineers
- Product Maintenance Manager

It is important to ensure that members of Change Control Board represent both vendor and customer side and individuals have knowledge in different domains (technical, business etc.)

7.3.1 Impact Analysis (K3)

To evaluate the possible effects of proposed change in order to make a reasonable decision on implementing or not the change, there should be impact analysis performed. Change Impact Analysis (IA) can be defined as:

- Identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change [Bohner, S.A. and R.S. Arnold]
- The evaluation of the many risks associated with the change, including estimates of the effects on resources, effort, and schedule [Pfleeger, S.L. and J.M. Atlee]

Both the design details and risks associated with modifications are critical to perform IA within change management processes.

The purpose of Impact Analysis is to identify the consequences of implementing requested change in the context of the whole software system or business process. These consequences can be expressed as a list of risks related to the change and usually includes estimations on cost, effort and schedule of the change implementation.

Impact Analysis techniques can be classified into three types [Kilpinen, M.S] (K3):

- Traceability – analyzing links between requirements, specifications, design elements and tests cases to determine the scope of a change.
- Dependency – analyzing links between parts, variables, logic, modules to determine the consequences of a change.
- Experiential - impact of changes is determined using expert knowledge.

Supporting techniques are (K3):

- Review meetings
- Informal team discussions
- Individual engineering judgment

The result of Impact Analysis should be communicated to stakeholders involved in change processing and then decided by Change Control Board on approving, rejecting or deferring the change.

7.3.2 Change Initiation (K2)

After a Change Request has been raised, the change is categorized and prioritized based on the information available (usually the information obtained as a result of the Impact Analysis of the change). Then the change should be discussed to establish the next steps. This is done during a Change Initiation Review.

Change Initiation Review (K2)

Change Initiation Review is a set of guidelines and templates providing the CCB with an initial checkpoint before approving a change to the production environment.

The purpose of a Change Initiation Review is to:

- Assess the key attributes of a change (priority, risk, effort) specified in the approval process against standards, policies, and quality metrics
- Evaluate the completeness of preliminary rollout, training, support and cost/benefit plans
- Consider whether to accept the change for implementation and deployment
- Consider whether to approve plans for operating and supporting the change (like training)
- Consider whether to approve the plans required for the readiness of the target production environment to operate and support the deployed change

The Change Initiation Review results in a Go/No-go decision about whether to approve the Change Request and initiate the development of the solution.

If the decision is “Go” the change continues into development. In case of “No-go” decision, the change is returned (with justification of the rejection) to the change initiator for verification, more information or rework.

7.3.3 Change Prioritization (K2)

The Change Requests must be prioritized. Prioritization supports the CCB in the process of reviewing changes and deciding on implementation. Change Requests may be prioritized based on the Kano model for quality factors. After Change Requests are prioritized, they are ready for review by the Change Control Board.

For product organizations, the prioritization has to be aligned with the Roadmap.

Kano Model (K2)

The Kano Model of Customer satisfaction divides product attributes into three categories: threshold, performance and excitement (sometimes called also as obvious, required and surprise). A competitive product meets basic attributes, maximizes performances attributes, and includes as many “excitement” attributes as possible taking into account the available budget. The Kano Model may be used in the Change Prioritization process to divide the Change Request into three categories, known also as: “must have”, “should have” and “nice to have”.

The procedure of Change Prioritization includes the following steps:

- Examining the Requirements List – all requirements affected by the Change Request should be examined. It must be stated if the requirements affected by the change are in the threshold, performance or excitement categories of the Kano model. In case the Change Request is adding a new requirement, the Kano model’s category should be determined for the new requirement.
- Establishing the priority – the priority should be determined for each new Change Request. Usually the priority is defined as:
  - High (for the “threshold” requirements created or affected by the Change Request)
  - Medium (for the “performance” requirements created or affected by the Change Request)
  - Low (for the “excitement” requirements created or affected by the Change Request)
- Expediting the Change Request (optional step) – expediting a Change Request may be done if the change is critical for the users or has to correct serious defect(s) in production.

In that case the priority should be set to Expedite and the Change Request is reviewed immediately by the CCB.

It is important to remember that the Expedite priority should be used rarely and only if really needed as expedited Change Requests can decrease the overall productivity.

- Reassigning the Change Request – the prioritized Change Requests should be reassigned to the release manager. Then they are ready for review by the Change Control Board.

7.3.4 Change Consolidation (K2)

Change Consolidation (K1)

Change Consolidation is the consolidation of changes from multiple sources into one change.

The purpose of Change Consolidation (K2)

The goal of consolidating the changes is to:

- Managing changes requesting similar features but coming from different stakeholders
- Managing changes related to the same area of the functionality/module in order to avoid conflicts (as one change requested to a specific functionality may stay in conflict in other changes submitted for the same functionality)
- Combining a number of minor changes requesting the same modification in the whole application (like changing the page header's colors, logos etc.)

7.4 Review methods (K3) — 60 minutes

Terms

Inspection, Follow-up, Formal Review, Informal Review, Kick-off, Manager, Moderator, Review, Review Meeting, Scribe, Technical Review, Walkthrough

Background

A review is a type of static testing. Reviewers are able to find defects, inconsistencies and gaps in requirement specifications by directly examining documents. The base for review are source documents (like requirement specification), standards to which the documentation must adhere and other references (documented user's needs, feedback from stakeholders).

The different types of reviews vary from informal, characterized by no written instructions for reviewers, to systematic, characterized by including team participation, documented results of the review, and documented procedures for conducting the review. The formality of a review process is related to factors such as the maturity of the development process, any legal or regulatory requirements or the need for an audit trail.

The way a review is carried out depends on the agreed objectives of the review (e.g., find defects, gain understanding, educate testers and new team members, or discussion and decision by consensus).

Reviews should be conducted as soon as the relevant documents are available. Reviewers must be provided with the documentation to be reviewed in adequate time to allow them to become familiar with the contents of the document before starting the official review process. This is often neglected and the documentation is provided too late. In result, the documentation is reviewed briefly and incompletely and not all important issues are found.

A review can lead to the possible results:

- The document is declared as ready to use unchanged or with minor changes
- The document must be changed but a further review is not necessary
- The document must be extensively changed and a further review is necessary

A typical formal review has the following main activities (K3):

- Planning:
    - Defining the review criteria

    - Selecting the personnel
    - Allocating roles
    - Defining the entry and exit criteria for more formal review types (e.g., inspections)
    - Selecting which parts of documents to review
- Kick-off:
    - Distributing documents to the review participants
    - Explaining the objectives and process of review and purpose and contents of relevant documents to the participants
    - Checking entry criteria (for more formal review types)
- Individual preparation
    - Preparing for the review meeting by reviewing the document(s) by individual participants
    - Noting potential defects, questions and comments (performed individually by each of participants)
- Review meeting - examination, evaluation, recording of results :
    - Discussing or logging, with documented results or minutes (for more formal review types)
    - Noting defects, making recommendations regarding handling the defects, making decisions about the defects
    - Examining/evaluating and recording during any physical meetings or tracking any group electronic communications
- Rework:
    - Fixing defects found (typically done by the author; in case of defects or gaps requiring requirements analysis or corrections, it may require input from other persons, like business stakeholders)
    - Recording updated status of defects (in formal reviews)
- Follow-up:
    - Checking that defects have been addressed and resolved
    - Gathering metrics
- Checking on exit criteria (for more formal review types)

<u>Roles and Responsibilities (K2)</u>

A typical formal review will include the roles below:

- Manager: decides on the execution of reviews, allocates time in project schedules and determines if the review objectives have been met.

- Moderator: the person who leads the review of the document or set of documents, including planning the review, running the meeting, and following-up after the meeting. If necessary, the moderator may mediate between the various points of view and is often the person upon whom the success of the review rests.
- Author: the writer or person with chief responsibility for the document(s) to be reviewed.
- Reviewers: individuals with a specific technical or business background (also called checkers or inspectors) who, after the necessary preparation, identify and describe findings (e.g., defects) in the product under review. Reviewers should be chosen to represent different perspectives and roles in the review process, and should take part in any review meetings.
- Scribe (or recorder): documents all the issues, problems and open points that were identified during the meeting.

Types of Reviews (K3)

A single document may be the subject of more than one review. If more than one type of review is used, the order may vary. The main characteristics, options and purposes of common review types are:

- Informal Review
- Walkthrough
- Technical Review
- Inspection

Walkthroughs, technical reviews and inspections can be performed within a peer group.

Success Factors for Reviews (K2)

Success factors for reviews include:

- Each review has clear and predefined objectives
- The right people for the review objectives are involved
- Defects found expressed objectively
- The review is conducted in an atmosphere of trust; the outcome will not be used for the evaluation of the participants
- Review techniques are applied that are suitable to achieve the objectives and to the type and level of software work products and reviewers
- Checklists or roles are used if appropriate to increase effectiveness of defect identification

- Training is given in review techniques, especially the more formal techniques such as inspection
- Management supports a good review process (e.g., by incorporating adequate time for review activities in project schedules)
- There is an emphasis on learning and process improvement

7.5 Prototyping and simulation methods (K2) — 40 minutes

Terms

Business Process Simulation, Dynamic Systems Development Method, Extreme Prototyping, Evolutionary Prototyping, Horizontal Prototype, Incremental Prototyping, Operational Prototyping, Prototyping, Simulation, Vertical Prototype, Throwaway Prototyping

Background

Prototyping (K1)

As explained in Chapter 3. Risk Reduction through Prototyping (K3) prototyping is the activity of creating prototypes of software application, its components or modules or single screens.

The benefits of prototyping in context of quality may be the following (K2):

- Help in validating the design – team members may check if the proposed solution is what the stakeholders need and what they described in requirements.
- Help in requirements documentation – prototyping not only makes the documenting process easier (as the writer can base the document on visible and testable prototype: this is especially useful in describing user interface requirements and aspects related to the navigation), but allows to verify the correctness and ability to implementation of already defined requirements. Furthermore, when using prototyping very often new requirements are discovered, not identified in initial requirements elicitation process.
- Requirements Engineer and other members of development team can get valuable feedback from the stakeholders and improve the design according to the expectations. Stakeholders may check if the proposed solution satisfies their needs and expectations.

Dimension of prototypes (K2)

Prototyping can be done horizontally or vertically.

- Horizontal prototypes give a broad view on the system, focusing rather on user interaction, than the functionality. They are often used in creating user interface prototypes. Horizontal prototypes can be used to gathering confirmation of user interface requirements and system scope and to develop initial estimates of development time, cost and effort.
- Vertical prototype requires more complete preparation of a single component, subsystem or function. It can be used to obtain detailed requirements for a given function with the

purpose to collect information regarding data volumes and system interface needs, network sizing and performance engineering.

Types of prototyping (K3)

There are many variants of prototyping. Most important are:

- Throwaway prototyping
- Evolutionary prototyping
- Incremental prototyping
- Extreme prototyping

Throwaway prototyping (K3)

Also called close-ended prototyping. Throwaway or Rapid Prototyping create a model that will be rather discarded than become a part of the final solution. As the preliminary requirements elicitation is accomplished, a simple working model of the system is constructed to visually show the users what their requirements may look like when they are implemented into a finished system.

The main advantage of Throwaway Prototyping is that it can be done quickly and the stakeholders may get quick feedback regarding their requirements. And if so, they are also able to change or refine the requirements in early stages of the system development.

Another advantage of Throwaway Prototyping is the ability to construct interfaces for some usability testing.

In Throwaway Prototyping the prototype is constructed in the following steps:

- Define preliminary requirements
- Design the prototype
- Stakeholders examines/uses the prototype and specify new requirements
- Repeat if necessary
- Define the final requirements
- Develop the real products

Evolutionary prototyping (K3)

The main goal of Evolutionary Prototyping (also known as breadboard prototyping) is to build a very robust prototype and constantly refine and rebuild it. The reason for this is that the Evolutionary prototype will form the core of the new system, and the improvements and further requirements will be built [Alan M. Davis].

This approach allows to modify already created prototype adding new features or introducing changes that couldn't be identified during the requirements and design phase.

In Evolutionary Prototyping, developers can focus themselves to develop parts of the system that they understand instead of working on developing a whole system.

To minimize risk, the developer does not implement poorly understood features. The partial system is sent to customer sites. As users work with the system, they detect opportunities for new features and give requests for these features to developers. Developers then take these enhancement requests along with their own and use sound configuration-management practices to change the software-requirements specification, update the design, recode and retest.

Incremental prototyping (K3)

The final product is built as separate prototypes. At the end the separate prototypes are merged in an overall design.

Extreme prototyping (K3)

This approach requires breaking the development into three phases. Each phase is based on the preceding one. The first phase is creation of a static prototype (most often consisting of HTML pages). Then the screens are programmed to be fully functional. This is done in the second phase. In the last phase programmers add services.

Prototyping methods (K3)

Several prototyping methods can be applied in Requirements Engineer's work; two of them are introduced below:

- Dynamic systems development method
- Operational prototyping

Dynamic Systems Development Method (K2)

DSDM [DSDMC] is a framework for developing business solutions based on prototyping. In DSDM prototype may be a diagram or a business process, in some cases also a production system. DSDM prototypes are incremental and evolve from simple forms (for example, an UML diagram) into more complex ones.

DSMS recommends four categories of prototypes (K1):

- Business prototypes – with the purpose to design the solution.
- Usability prototypes – used to define and demonstrate usability and look and feel of the user interface design.
- Performance and capacity prototypes – used to predict and demonstrate how the system will perform non-functional requirements.
- Capability/technique prototypes – used to develop, demonstrate, and evaluate a design approach or concept.

DSDM lifecycle of a prototype (K1):

- Identify prototype
- Agree to a plan
- Create the prototype
- Review the prototype

Operational prototyping (K2)

The main concept of Operational Prototyping is to integrate throwaway and evolutionary prototyping with conventional system development. According to the author of this approach, Operational Prototyping provides the best results of throwaway and evolutionary prototyping and the traditional development. In this method programmers build the evolutionary prototype only with well-understood requirements and use throwaway prototyping to experiment with the poorly understood requirements. This allows to proceed on well described and clear requirements and in the meantime to clarify the less understood ones.

Simulation (K2)

Business Process Simulation is a technique allowing to simulate the execution of business processes and their parameters over time based on process models. Such models must represent not only the specific elements of the business process, but its attributes as well (execution time, resource usage, cost). Running such simulation allows to check how the process is realized, what resource usage on every step of the process is, where potential bottleneck and areas of instability are.

Business Process Simulation helps to understand, analyze and design (or re-design) business process models with respect to performance metrics such as throughput time, cost or resource utilization. Using simulation allows to evaluate and compare the (re)designed processes and decide on the best choice to implement within the organization.

Simulation may be used whenever there is a need for optimizing the business processes in an organization. As the processes are becoming more and more complex, optimization is an important element of increasing the organization's performance. Changing the existing processes in intuitive way may lead to unexpected negative results and lower process performance instead of reaching the designer goal. Simulation provides quantitative estimates of the impact that a new process design is likely to have on process performance, therefore most reasonable decision on the choice for the best design can be made.

<u>BPS Procedure (K2)</u>

The simulation of business processes is performed in several steps:

- Mapping the business process onto a process model
- Identification of the sub processes and activities
- Creating the control flow definition (determining and describing the connectors linking the different parts of the process)
- Identification of the resources are assigning them to the activities
- Defining performance characteristics (realization time, resource utilization)

After completing these activities, the simulation may be run. To ensure better and more reliable results, the simulation should be executed for several sub runs, each of sufficient run length.

Simulation is run in specific tool. Most tools show an animated picture of the process flow or real-time fluctuations in the key performance measures. This way it is possible to observe the sequence of performing specific actions within the process.

7.6 Different validation techniques (K2) — 60 minutes

Terms

Checklists, Formal validation, Group presentation, Inspection, Prototyping, Review Testing, Tracing Approaches, User Manual

Background

The following techniques may be used for requirements validation:

- Group presentation
- Prototyping – Usability and functionality testing
- Tracing approaches
- Testing
- User manual writing
- Formal validation
- Reviews and inspections
- Checklists

<u>Prototyping (K)</u>

Prototyping is an effective method to demonstrate the proposed implementation of stakeholders' requirements. Using prototypes helps stakeholders discover problems and validate requirements. As the prototype is more accessible and readable than specification, validation based on prototyping usually gives good results.

Validation process in case of using prototypes includes the following steps:

- Prototypes testers selection
- Test cases and test scenarios preparation
- Test execution
- Registration/documentation of problems found

It is important to plan the test set carefully and with proper objectives. Tests have to cover the requirements and have to be a guideline for the testers.

Tracing (K2)

Tracing allows to perform various checks – that all requirements elicitation notes are covered or checking stated goals against tasks, features, requirements. Common tracing technique is building Traceability Matrix.

Using tracing allows to ensure all requirements have been taken into consideration when creating detailed system specifications and implemented according to those specifications.

Testing (K2)

Validation can also be performed using testing. Each requirement should be testable - it should be possible to create test cases checking whether or not a requirement has been met. If the requirement is not testable – it means the requirements is not well defined and must be refined.

Test cases based on requirements should be traced to requirements. This allows to ensure proper requirements coverage.

Requirements tests is one of the most effective technique as any missing or ambiguous information in the requirements description may make it difficult to define test case or test scenario.

Notice: some software development approaches (Agile methods) start from test cases (Test Driven Development). In this approach, testing precedes writing the code.

Notice: some requirements are difficult to validate using testing – like non-functional requirements such as reliability. In this case, additional validation techniques should be used.

Writing User Manual (K2)

Writing user manuals may be a validation technique. It forces a detailed look at requirements, especially those related to usability. It may be helpful to identify the areas of the application that may be problematic to the end user and would not satisfy his usability needs.

User manual usually contains the following information

- Explanation how to install (if applicable) and get started with the system
- Description of the functionality and how it is implemented (including detailed usage scenarios with screens)
- Description of possible exceptions

Formal validation (K2)

The purpose of formal validation is to check if a formal specification has certain desirable properties like completeness, consistency etc.

Review and inspections (K2)

For requirement validation purpose, reviews and inspections focus on checking the compliance of the requirements specifications with the real needs and expectation of the stakeholders, especially the customer. Therefore, reviews and inspections must involve selected group of stakeholders, including the customer representative.

Checklists (K2)

Checklists allow to validate the requirements according to specified criteria described in a form of list of questions/issues. The person performing the validation is checking the specification following previously prepared checklist and notes the results directly on that list. There are some ready checklists available for free in the internet but it is important to adjust the content to the needs and assumptions of specific project.

# 8 Quality Assurance (K4)	120 minutes

Learning Objectives for Advanced Level of Requirements Engineering
The objectives identify what you will be able to do following the completion of each module.

8.1 Quality Assurance Activities (K3)

LO-8.1.1	Explain, giving examples, common Quality Assurance activities (K3)
LO-8.1.2	Understand and explain the Requirements Engineering role in Quality Management Systems (K2)
LO-8.1.3	Describe which elements of Requirements Engineering can improve the quality of development process (K4)
LO-8.1.4	Describe common standards and programs for quality management (K2)
LO-8.1.5	Explain goals and activities of Audit Techniques for Requirements Engineering (K2)
LO-8.1.6	Explain the purpose and available techniques of Process Improvement (K2)
LO-8.1.7	Describe and use, in given scenario, main elements of Process Improvement work (K3)
LO-8.1.8	Describe common metrics in Requirements Engineering (K2)
LO-8.1.9	Explain and provide solutions of measurement in Requirements Engineering Process (K3)

8.2 Quality Assurance through Testability (K3)

LO-8.2.1	Explain what Acceptance Criteria are (K2)
LO-8.2.2	Build Acceptance Criteria for a given scenario (K3)
LO-8.2.3	Use defined Acceptance Criteria to verify the quality of given requirement (K4)

8.1 Quality Assurance Activities (K3) — 120 minutes

Terms

Audit, CMMI, ISO 9000, ISO 9001, Metric, Process Improvement, Requirements Development (RD), Quality Management System, Requirements Management (REQM), Requirements Management Maturity Model, TickITplus

8.1.1 Requirements Engineering role in Quality Management Systems (K2)

Whilst Quality Control performs a valuable function in the attainment of quality, the operational techniques and activities employed cannot assure quality alone.

Quality Assurance (K2)

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."

The key words in this definition are that the actions taken are "planned and systematic" and that these actions "provide adequate confidence" that the desired level of quality will be achieved. These actions include those operational techniques and activities used to fulfill requirements for quality. Quality Control forms an essential part of the measures taken to achieve Quality Assurance.

ISO 9000:2000 defines Quality Assurance defined as:

"Part of quality management focused on providing confidence that quality requirements will be fulfilled."

The "confidence" comes from knowing that all necessary preventative measures have been taken before the process is started.

One of the main concepts of Quality Assurance is to ensure the best possible quality by realizing specified activities at each stage of the project, what would allow to identify potential problems as soon as possible. The role of Quality Assurance is to ensure that processes are appropriate and correct (resulting in providing desired outputs and performed according to the process requirements) and later implemented according to plan. Relevant measurement processes measure the level of quality. All these activities can be defined in Quality Assurance Plan or Quality Plan.

The major activities of Quality Assurance are the following :

- Establishing a Quality Management System and processes for Requirement Engineering and training people in it
- Performing Audits of Requirement Engineering and following up corrective actions
- Performing measurements and analysis of Requirement Engineering Processes
- Establishing and running Process improvements program to Improve Requirement Engineering processes

<u>Quality Management System (K2)</u>

Quality Management Systems, based on the concept of prevention, provides confidence to both suppliers and customers that defined requirements will be met. This implies that having a management system, processes, procedures and tools together with skilled and properly trained personal is a very important part of assuring quality in Requirement Engineering.

A Quality Management System (QMS) is an organizational structure, procedures, processes and resources needed to implement Quality Management.

Elements of a Quality Management System

- Organizational Structure
- Responsibilities
- Methods
- Data Management
- Processes
- Resources
- Customer Satisfaction
- Continuous Improvement
- Product Quality

Requirements Engineering contributes to most of elements of a Quality Management System

- Responsibilities (listing the role and responsibility of Requirements Engineer, System Analyst, Business Analyst)

- Methods (methods and techniques of Requirements Analysis, Design, Modeling and Documentation)
- Data Management (requirements documentation management)
- Processes (processes of Requirements Analysis, Design, Modeling and Documentation, Requirements Quality Assurance)
- Resources (tools and human resources related to the Requirements Engineering)
- Customer Satisfaction (quality of Requirements Engineering as a key factor for ensuring customer satisfaction by meeting the needs and expectations regarding the software)
- Product Quality (directly resulted from the fact that the quality of a product is determined by the quality of requirements)

ISO 9000 (K2)

The ISO 9000 family of standards relate to Quality Management Systems and are designed to help organizations ensure they meet the needs of customers and other stakeholders.

ISO 9000 deals with the fundamentals of quality management, including the eight management principles on which the family of standards is based.

ISO 9001 deals with the requirements that organizations wishing to meet the standard have to comply with.

ISO 9001:2008 Quality Management Systems — Requirements (K2)

The standard specifies requirements for a quality management system where an organization:

- needs to demonstrate its ability to consistently provide products that meet customer requirements, and
- aims to enhance customer satisfaction through the effective application of the Quality Management System.

These requirements are structured into the following 5 clauses:

- **Quality management systems** including general requirements and requirements on documentation.
- **Management responsibilities** defining requirements for management commitment and review, customer focus and policy as well as planning and responsibilities, authorities and information.

- **Resource management** including requirements on provision of resources, human resources, infrastructure and work environment.
- **Product realization** including requirements on processes for planning, customer relations, design and development, purchasing, product and service provisioning and control of monitoring and measuring devices
- **Measurement, analysis and improvement** including requirements on monitoring and measurement, control of nonconforming product, analysis of data and improvement.

There are six compulsory documents specified by the standard:

- Control of Documents (4.2.3)
- Control of Records (4.2.4)
- Internal Audits (8.2.2)
- Control of Nonconforming Product / Service (8.3)
- Corrective Action (8.5.2)
- Preventive Action (8.5.3)

The key documents requires by ISO 9001:2008 are a Quality Policy and Quality Manual.

Requirement Engineering usually contributes to:

- Quality Policy (statements related to the satisfying of the customer's needs and expectations)
- Section 7: Product Realization (Requirements Engineering's practices used in the development process, quality attributes etc.)
- Section 8: Measurement, analysis and improvement (measuring on Requirements Engineering's process and products)

Specifically, the following sections are closely related to Requirement Engineering:

7.2.1 Determination of requirement related to the product

7.2.2 Review of requirement related to product

7.3.2 Design and development inputs

<u>TickIT and TickIT plus (K2)</u>

TickIT is a certification program for Quality Management in software development. The TickIT guide provides information allowing to understand and apply ISO 9001 in the IT industry together with detailed information on how to implement a Quality Management System and its expected structure and content relevant to software activities. The TickIT guide also helps to define appropriate measures and metrics.

The original TickIT scheme has been updated to become TickITplus.

TickITplus is both an Improvement Tool and an ISO 9001 based IT accredited certification scheme. TickITplus extends the existing TickIT Scheme combining industry best practice with International IT standards. It uses the following standards:

- ISO 9001:2008
- ISO/IEC 15504
- ISO/IEC 12207

Amongst others, TickITplus provides a defined process model covering the whole range of IT activities, not just software development and a revised qualification and training structure for both Assessors and Practitioners.

TickITplus is built around the ISO/IEC 15504 standard – IT Process Assessment. There are five levels of the assessment:

- Foundation – this is the normal entry level and requires a process model to be defined and verified, but there is no direct process assessment.
- Bronze – this equates to level 2 (the Managed level in ISO/IEC 15504), and ensures the processes are operated with planned, monitored and adjusted management.
- Silver – this equates to level 3 (the Established level in ISO/IEC 15504), and ensures that processes are capable of achieving their outcomes in terms of definition and deployment.
- Gold – this equates to level 4 (the Predictable level in ISO/IEC 15504), ensuring that processes operate within predicted parameters.
- Platinum – this equates to level 5 (the Optimizing level in ISO/IEC 15504), and ensures that quantified measures and improvements are applied to key processes.

8.1.2 Audit Techniques for Requirements Engineering (K2)

Audit (K1)

Audit is an evaluation of a person, organization, system, process, enterprise, project or product. It is the most formal review technique.

Audit process (K2)

The Audit process consists of the following phases:

- Establishing the terms of the engagement – addressing the responsibility (scope, independence, deliverables), access to information) and accountability of the auditor.
- Preliminary Review – gathering organizational information for creating the Audit Plan.
- Evaluating risks – assessing the organization's business risks in order to plan the Audit.
- Planning the Audit – this will help to ensure the audit is conducted in an effective and efficient manner. When creating the audit plan, the results of the risk assessment process should be considered.
- Considering information from internal control – the auditor should consider information from previous audits, the assessment of risks and the complexity of the organization's operations and systems.
- Performing audit procedures – usually supported by checklists based on standard guidelines and recommendations. The results of performing each of the audit activity should be documented. This step is known also as "fieldwork".
- Preparing the Audit Report – after completing the audit procedures and evaluating the results, the auditor creates an Audit Report (unqualified or qualified) based on the findings.
- Follow-up – defects and deviation from the planned process / products found during the audit should be presented to the management. Possible resolution of the problems should be discussed and agreed upon. Corrective actions should be undertaken. The management should provide the timeframes for fixing activities. Once the deadline is met, the results of the corrective actions will be reviewed.

Audit Plan (K2)

The Audit Plan describes the objectives of the audit and steps to be performed to ensure all of the important issues in the audit are covered. Usually the Audit Plan includes:

- Purpose of the audit

- References – listing of relevant standard(s), guides etc.
- Potential audit risks
- A basic framework for resource allocation (together with the resource required to the realization of the audit)
- Audit procedures to be performed

In the Requirement Engineering audit can be used to:

- Assess Requirement Engineering process itself (compliance with relevant standards etc.)
- Evaluate the quality of Requirement Engineering's products (requirements specification, models etc.)
- Identify any deviations from the planned realization of the Requirements Engineering process

The following items can be a subject of an audit:

- Products of the Requirement Engineering process (requirements specifications, models (like UML))
- The process – effectiveness of the activities, techniques, tools used to perform the Requirement Engineering and their compliance with specified standards or declared organization's approach to the Requirement Engineering process
- The customer – the customer's feedback as an information regarding the quality of the Requirements Engineering process, activities and products offered by the software vendor

Types of audits (K1)

The following types of audits can apply to the Requirements Engineering:

- Selective, where a sample is taken of handling of requirements
- Upstream, where the audit starts with the validation of a requirement an goes backwards to check if there is a full traceability back to the requirement, including changes
- Downstream, following an initial requirements or change request from initial analysis, design, development and checking all verification and validations as well as the traceability

An Audit may be internal (performed by the organization's employees) or external (conducted by a third party, usually a professional auditor).

8.1.3 Process Improvement work (K2)

Increasing the quality characteristics of specified process is the goal of Process Improvement. Good Requirements Engineering results in improving the overall production process, as it allows to:

- Speed up the implementation efforts by eliminating waste of time for clarifying and explaining requirements
- Reduce the number of defects caused by errors in requirements specification or/and misunderstanding of requirements documentation resulting from its low quality
- Increase the quality of testing by providing well described, precise and testable requirements
- Make the acceptance testing easier and more reliable as the basis for such testing are mostly requirements

Therefore the role of Requirements Engineering in improving the development process is very important. To improve the Requirements Engineering itself the following actions can be taken:

- Applying standards and good practices related to Requirements Engineering
- Educating personnel about the meaning and role of Requirements Engineering
- Designing and introducing organization's approach to Requirements Engineering
- Using carefully selected and matching to the organization's and project's needs tools supporting r Requirements Engineering
- Introducing Quality Control into Requirements Engineering activities (like audits, reviews etc.)
- Knowledge sharing between Requirements Engineering professionals within an organization (internal trainings, workshops, lessons learned and experience exchange, common knowledge base etc.)
- Using lessons learned to improve the future projects

Improvement can apply to each of the Requirements Engineering activities:

- Identification of Requirements – this activity may be improved to collect customer's requirements in more effective way. In order to do so, the organization may introduce some techniques allowing to ensure the requirements are gathered faster, are complete and agreed upon most stakeholders. Some of such techniques are interviews, brainstorming, initial prototyping, using Personas, scenarios.

- Requirements Analysis – identified requirements have to be analyzed and detailed. Improving the process of Requirements Analysis ensures that the risk of overlooking important aspects of requirements is lowered. Requirement Analysis should involve the customer representative (to verify the results of work as they appear), development team (to check the technical feasibility of detailed requirements and provide early feedback) and the Quality Assurance team – to ensure all requirements and testable.
- Specification of Requirements – to improve Specification of Requirements the organization may introduce reviews, checklists and apply specified Requirements Engineering standard(s). Common practice is releasing the draft specification for internal technical (development) and Quality Assurance review. This way most of mistakes and gaps may be captured and corrected in the next version of the specification.
- Tracking and Control of Requirements – to improve the quality control, it is necessary to collect measurements and deficiencies, make corrections and train people, but also to analyze trends and other business results in order to improve the way of working.
- Quality Assurance – related to the fact, that having a Quality Management System and improving it is becoming more and more popular. The improvement is often run in the processes. More organizations are locking into models such as TickITplus, Spice and CMMI to assist in the improvement work.

The processes in the different improvement models that are related to Requirement Engineering are:

ISO 15504

- ACQ1 Acquisition preparation BP1-3
- ENG1 Requirements elicitation BP1-6
- ENG2 System Requirements Analysis BP1-6
- ENG4 Software Requirements Analysis BP1-6

CMMI

- 2 Requirements Management
- 3 Requirements Development

TickITplus

- AGR 1 Acquisition and Contract Management
- TEC10 Stakeholder Requirement Definition
- TEC11 Requirements Analysis

TickITplus

Improvements are integral part of TickITplus. The way they are monitored and controlled within the scheme depends on the different maturity grades. For example, at the Foundation level the Improvements Plan is a requirement only; at higher grades the contents of this plan form a planned activity within assessments. At the Platinum level additional high maturity processes dealing with quantitative analysis and improvements are defined

ISO 15504-2 requires a Process Reference Model to exist in order for a capability assessment to be completed. However, it would not be possible to define a single Process Reference Model which could satisfy all potential organizational types. For this reason, TickITplus has created a Base Process Library from which organizations select the most applicable processes to create an organizational specific Process Reference Model suited to their specific activities. The processes in BPL which are part of Requirement Engineering is TEC.10 and TEC.11.

TEC.10 Stakeholder Requirement Definition

To define the requirements of the products and services expected by the customer.

Base Practices

TEC.10.BP.1 Engage Requirements Stakeholders

TEC.10.BP.2 Develop Stakeholder Requirements

TEC.10.BP.3 Validate Stakeholder Requirements

TEC.10.BP.4 Manage changes to Stakeholder Requirements

TEC.11 Requirements Analysis

The customer needs and other stakeholder requirements are analyzed and interpreted into structured system requirements.

The organization considers its own product development strategy in light of stakeholder requirements.

System requirements are reviewed and maintained under configuration management.

Base Practices

TEC.11.BP.2 Estimate System Requirements Size

TEC.11.BP.3 Manage System Requirements

TEC.11.BP.4 Manage Changes to Requirements

CMMI and Requirement Engineering (K2)

Capability Maturity Model Integration (CMMI) is a process improvement approach that provides organizations with the essential elements for effective process improvement. CMMI is a trademark owned by Software Engineering Institute of Carnegie Mellon University.

CMMI helps "integrate traditionally separate organizational functions, set process improvement goals and priorities, provide guidance for quality processes, and provide a point of reference for appraising current processes." [after SEI]

CMMI best practices are published in a form of “models” each of which addresses a different area of interest. The current release of CMMI, version 1.3 describes the following models:

- CMMI for Development (CMMI-DEV). It addresses product and service development processes.
- CMMI for Acquisition (CMMI-ACQ). It addresses supply chain management, acquisition, and outsourcing processes in government and industry.
- CMMI for Services (CMMI-SVC). It addresses guidance for delivering services within an organization and to external customers.

Maturity Levels (K2)

CMMI has five maturity levels. Maturity level ratings are awarded for levels 2 through 5.

- Maturity Level 1 - Initial
- Maturity Level 2 - Managed (REQM - Requirements Management)
- Maturity Level 3 - Defined (RD - Requirements Development)
- Maturity Level 4 - Quantitatively Managed
- Maturity Level 5 - Optimizing

The following areas of CMMI are related to Requirements Engineering:

- Requirements Development (RD) – the purpose of RD is to produce and analyze customer, product, and product component requirements. Specific Practices by Goal are:
    - SG 1 Develop Customer Requirements
        - SP 1.1 Elicit Needs
        - SP 1.2 Develop the Customer Requirements
    - SG 2 Develop Product Requirements
        - SP Interface Requirements
    - SG 3 Analyze and Validate Requirements

        - SP 3.1 Establish Operational Concepts and Scenarios
        - SP 3.2 Establish a Definition of Required Functionality
        - SP 3.3 Analyze Requirements
        - SP 3.4 Analyze Requirements to Achieve Balance
        - SP 3.5 Validate Requirements

- Requirements Management (REQM). The purpose of REQM is to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products. Specific Practices by Goal are:
    - SG 1 Manage Requirements
    - SP 1.1 Obtain an Understanding of Requirements
    - SP 1.2 Obtain Commitment to Requirements
    - SP 1.3 Manage Requirements Changes
    - SP 1.4 Maintain Bidirectional Traceability of Requirements
    - SP 1.5 Identify Inconsistencies Between Project Work and Requirements

Requirements Management Maturity Model (K2)

The Requirements Management Maturity Model deals with the increasing levels of formality and sophistication in eliciting and managing requirements.

The model comes in five levels [Rational Software].

1. Chaos
2. Written requirements
3. Organized
4. Structured
5. Integrated

General maturity model (K2)

In general can you describe that Requirement Engineering on the different majority models means:

Level 1. Problem with Requirements is normal, Initial level

- A process for requirement engineering is missing; there are often problems with unsatisfied customers and unresolved requirements, the process does not use methods and is depending on individuals.

Level 2. Requirements Specifications of better value, Repeatable Level

- Standards for Requirement documents and Requirement expressions exist as well as procedures for Requirement Management, there are some tools and techniques in use.

Level 3. Requirement Engineering process is established, Defined level

- There is a process model which is build on experiences and established methods. An improvement work is done based on independent evaluations and new methods and tools.

8.1.4 Measurement on Requirements Engineering Process (K3)

Some of metrics allowing to measure the Requirements Engineering are:

- The number of requirements by priority
- The number of requirements by complexity
- The number of dependent, linked or related requirements
- The number of use cases per module
- The number of interfaces to external systems

The Requirements Engineering may not only be measured in terms of the outputs (like requirements) but in the term of the process itself. The maturity of the process may be measured and evaluated (see: Maturity levels in previous section). The maturity may be measured using the following criteria:

- The organization of the Requirements Engineering process and its place in the software development process
- Compliance with relevant standards

- The number and type of Requirements Engineering's documents adjusted to the organization's needs
- Usage of tools supporting the Requirements Engineering process
- The experience and competences of the personnel (measured i.e. by certification)
- Planning – meeting deadlines, causes of delays etc.

The quality of Requirements Engineering can be measured using the following metrics:

- The number of issues found during specification reviews (by priority, severity)
- The number of issues found in requirements during implementation or testing (by priority, severity)
- The cost of fixing issues found in requirements in every project phase

8.2 Quality Assurance through Testability (K3) | 60 minutes

Terms

Acceptance Criteria, Testability

Background

Requirements engineering is closely connected to testing. Good test cases require good requirements that can be tested (The involvement of testers for the specification is therefore very important) (K2).

Acceptance Criteria (K2)

According to Prince2™ nomenclature, Acceptance Criteria, called also Success Criteria, are the standards required to satisfy the customer's quality expectations and gain the customer's acceptance of the final product. In other words, they are criteria put on specified function, component or any other item of the software product or other output of the project, that must be met to satisfy the customer and gain his acceptance of the product.

The Acceptance Criteria should be agreed upon by both sides – the vendor and the customer – before starting the project (they should be a part of contract documentation) and will form the basis for the Project Quality Plan. Every requirement must have at least one acceptance criterion. Such criteria are the basis for the Acceptance Testing.

Each of the criteria must be measurable and the means of measuring the criteria must be realistic and agreed.

Acceptance Criteria are not only related to the products of the project, but also to the project itself.

Some criteria that should be considered include:

- Deadlines
- Necessary skills, competences and experience of the personnel developing the product
- Repair times
- Development costs
- Running costs
- Levels of testing

- Testing coverage

Other useful general acceptance criterion is connected to the severity of the faults found in test.

When it comes to services the Acceptance Criteria are often described as Service Levels that are continues measured and contracted in Service Level Agreements (SLA).

There are a lot of different activities that can form part of the acceptance. Some of them are:

- Factory acceptance/FAT
- Site acceptance/SAT
- Field-test
- Parallel operation
- Transfers
- Guaranties

9 Tools (K4) | 120 minutes

Learning Objectives for Advanced Level of Requirements Engineering
The objectives identify what you will be able to do following the completion of each module.

9.1 Advantages of Tools (K2)

LO-9.1.1	Explain the role of tools in improving Requirements Engineering (K2)
LO-9.1.2	Describe common advantages of using different types of tools (K2)

9.2 Categories of Tools (K2)

LO-9.2.1	Explain the purpose of different categories of tools (K2)
LO-9.2.2	Describe basic tools categorization (K2)

9.3 Use of Tools (K3)

LO-9.3.1	Explain which factors are main drivers for the tool selection (K2)
LO-9.3.2	Select most suitable tool for resolving a problem in given scenario (requirement management, modeling, user interface designing) (K3)
LO-9.3.3	Describe goals and advantages of possible integration between RE tools and tools supporting other areas in a project (K3)

9.4 Practical Example for the Use of Tools (K3)

LO-9.4.1	Select adequate tool for modeling requirements in given scenario (K3)

9.1 Advantages of Tools (K2) — 20 minutes

Terms

Requirement Engineering Tools

Background

Tools for storage and administration of requirements facilitate Requirements Engineering. They can take on mechanical activities or ensure overview. It is thus possible to keep difficult static documents consistent and current. The selection of a tool must occur before the product is developed. Otherwise such situation may cause substantial problems (K2).

Tools may support the following activities in Requirements Engineering:

- Requirement identification and storage
- Requirements modeling (including prototyping)
- Requirement documenting (requirements specification creating)
- Defining and maintaining requirements traceability

The advantages of using tools may be the following (K2):

- Ensuring that all requirements are stored in one place and accessible for all involved stakeholders
- Supporting requirements traceability (of test cases etc.) and allowing to verify the relevant requirements coverage
- Allowing to manage requirements changes in easy way
- Improving the quality of requirements specification by forcing usage of defined document templates and modeling notation
- Time savings by automation on some activities (like generating complete specifications from the tool)

9.2 Categories of Tools (K2) — 20 minutes

Terms

Categories of Tools

Background

Categories of Tools (K2)

- Text processing, table calculations
- Requirements Elicitation Tools
    - Mind-mapping
- Modeling Tools
    - UML tools
    - SysML tools
- Prototyping Tools
- Requirements Management Tools
- Defect Management Tools
- Change Management Tools
- Project Management Tools

The cost for tools thereby varies greatly. The choice of a tool must therefore be made very carefully. A hasty choice can result in high costs (K2).

9.3 Use of Tools (K3) — 40 minutes

Background

Growing complexity and size of software systems, along with increasing number of requirements and their dependencies cause that using tools becomes necessary.

Tools can support works on each of the main activities of Requirements Engineering.

- Identification of Requirements
  - Requirements Management tools allowing to store requirements
  - Mind-mapping to support the process of gathering requirements from the stakeholders
  - Prototyping tools to create the initial proposals of the solution which would help the stakeholders to verify already identified requirements and discover new ones
  - Project Management tools to estimate the effort needed to complete Requirements Analysis and Specification and prepare the requirements development schedule
- Requirements Analysis
  - Requirements Management tools allowing to organize identified requirements into packages and create links and references between them
  - Prototyping tools to create the design of user interface elements and analyze requirements and their relationships implemented via this design
- Specification of Requirements (including modeling and documenting)
  - Modeling tools to design the requirements models
  - Prototyping tools to create the final design of user interface elements
  - Text editors to edit content of the requirements
- Tracking and Control of Requirements
  - Change Management tools to manage Change Requests
- Quality Assurance
  - Defect Management system to manage defects/issues discovered during requirements implementation or testing phases. A tool for Defect Management and for Change Management may be the same tool.

Different types and tools may be used on specific project. The selection depends on the following aspects:

- The size of the project (the number of requirements)
- The type of the project (development of completely new software, extending existing software, providing new functionalities and interfaces etc.)
- The complexity of requirements
- The applied level of requirement's formalization
- The development approach (different tools to be used on RUP projects and on Agile projects)
- The customer's requirements regarding the tools
- Knowledge and experience of the vendor's team and the customer (who should be familiar with a technique/notation used by a tool in order to understand the outcomes)

9.4 Practical Example for the Use of Tools (K3) — 40 minutes

The training provider should demonstrate the practical use of tools.

10 Literature

Beck, K.: *Extreme Programming*. Munich 2003

Beck, K.: *Extreme Programming Explained: Embrace Change*. Boston 2000

Beck, K.: *Test Driven Development. By Example*. Amsterdam 2002

Beck, K.: *Refactoring*: *Improving the Design of Existing Code*. Addison-Wesley Longman 1999

Boehm, B.: *Software Engineering Economics*. Englewoods Cliffs, NJ 1981

Bohner, S.A. and R.S. Arnold, Eds. *Software Change Impact Analysis*. Los Alamitos, California, USA, IEEE Computer Society Press 1996.

Bundschuh, M.; Fabry, A.: *Aufwandschätzung von IT-Projekten*. Bonn 2004

Cockburn, A.: Agile *Software Development*. Addison Wesley 2002

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

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

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

Newman, W.M. and Lamming, M.G.: *Interactive System Design*, Harlow: Addison-Wesley 1995

Davis A. M.: *Operational Prototyping: A new Development Approach*. IEEE Software, September 1992. Page 71

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

DeMarco, T. et al.: *Adrenalin-Junkies und Formular-Zombies – Typisches Verhalten in Projekten*. Munich 2007

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

DeMarco, Tom: *The Deadline: A Novel About Project Management*. New York 1997

Dorfman, M. S.: *Introduction to Risk Management and Insurance (9 ed*.). Englewood Cliffs, N.J: Prentice Hall 2007. ISBN 0-13-224227-3.

Dynamic Systems Development Method Consortium. See: http://na.dsdm.org

Ebert, Ch.: *Systematisches Requirements Management. Anforderungen ermitteln, spezifizieren, analysieren und verfolgen*. Heidelberg 2005

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

Heumann, J.: *The Five Levels of Requirements Management Maturity,* see: http://www.therationaledge.com/content/feb_03/f_managementMaturity_jh.jsp

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

IEEE Standard 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology

IEEE Standard 829-1998 IEEE Standard for Software Test Documentation

IEEE Standard 830-1998 IEEE Recommended Practice for Software Requirements Specifications

IEEE Standard 1012-2004: IEEE Standard for Software Verification and Validation

IEEE Standard 1059-1993: IEEE guide for software verification and validation plans

IEEE Standard 1220-1998: IEEE Standard for Application and Management of Systems Engineering Process

IEEE Standard 1233-1998 IEEE Guide for Developing System Requirements Specifications

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

ISO 9000

ISO/EIC 25000

ISO 12207

ISO 15288

ISO 15504

ISO 31000: RiskManagement - Principles and Guidelines on Implementation

IEC 31010: Risk Management - Risk Assessment Techniques

ISO/IEC 73: Risk Management – Vocabulary

Jacobsen, I. et al.: *The Unified Software Development Process*. Reading 1999

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

Kilpinen, M.S.: *The Emergence of Change at the Systems Engineering and Software Design Interface: An Investigation of Impact Analysis. PhD Thesis.* University of Cambridge. Cambridge, UK 2008.

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

Mangold, P.: IT-*Projektmanagement kompakt*. Munich 2004

McConnell, S.: *Aufwandschätzung für Softwareprojekte*. Unterschleißheim 2006

McConnell, S.: *Rapid Development: Taming Wild Software Schedules (1st ed*.). Redmond, WA: Microsoft Press. ISBN 1-55615-900-5, 1996

Paulk, M., et al: *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, MA 1995

Pfleeger, S. L.: *Software Engineering: Theory and Practice, 2nd edition*. Englewood Cliffs, NJ 2001

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

Pohl, K.: *Requirements Engineering. Grundlagen, Prinzipien, Techniken*. Heidelberg 2007

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

Spillner, A. et all: *Software Testing Foundations*. Santa Barbara, CA 2007

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

TickITplus: http://www.tickitplus.org/

V-Modell® XT: http://www.vmodellxt.de/

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

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

Wiegers, K. E.: *More About Software Requirements: Thomy Issues and Practical Advice*. Redmond, Washington 2006

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

11 Index