Dominik Häußer
Kim Lauenroth
Hans van Loenhoud
Anja Schwarz
Patrick Steiger

Handbook of Advanced Level Elicitation according to the IREB Standard

Education and Training for

IREB Certified Professional for Requirements Engineering

Advanced Level Advanced Level Elicitation

Version 1.0.3

July 2019

The compilation of this handbook was supported by

SOPHIST

Terms of Use

This Handbook, including all of its parts, is protected by copyright law. With the consent of the copyright owners and following copyright law, the use of the Handbook is permitted—unless explicitly mentioned it is not permitted. This applies in particular to reproductions, adaptations, translations, microfilming, storage and processing in electronic systems, and public disclosure.

Training providers may use this Handbook as a basis for seminars and training provided that the copyright holder is acknowledged and the source and owner of the copyright is mentioned. In addition, with the prior consent of IREB, this Handbook may be used for advertising purposes.

Any individual or group of individuals may use this Handbook as a basis for study, articles, books or other derived publications provided that the copyright holder is acknowledged and the source and owner of the copyright is mentioned.

Acknowledgements

This Handbook has been written by Dominik Häußer, Kim Lauenroth, Hans van Loenhoud, Anja Schwarz and Patrick Steiger.

Review by Birgit Penzenstadler. English Review by Gareth Rogers.

Approved for release on October 8, 2018 by the IREB Council upon recommendation of Thorsten Weyer.

We thank everybody for their involvement.

Copyright © 2017-2019 for this Handbook is with the authors listed above. The rights have been transferred to the IREB International Requirements Engineering Board e.V.

Table of Contents

Foreword

This *Handbook* complements the syllabus of the CPRE Advanced Level Elicitation.

This Handbook is intended for training providers who want to offer seminars or training on the CPRE Advanced Level Elicitation according to the IREB standard. It is also aimed at training participants and interested practitioners who want to get a detailed insight into the content of this advanced level module.

This Handbook is not a substitute for training on the topic. The Handbook represents a link between the Syllabus (which lists and explains the learning objectives of the module) and the broad range of literature that has been published on the topic.

The contents of this Handbook, together with references to more detailed literature, support training providers in preparing training participants for the certification exam. This Handbook provides training participants and interested practitioners an opportunity to deepen their knowledge of Requirements Engineering in an agile environment and to supplement the detailed content based on the literature recommendations. In addition, this Handbook can be used to refresh existing knowledge about the various topics of requirements elicitation, for instance after having received the certificate.

Suggestions for improvements and corrections are always welcome!

E-mail contact: info@ireb.org

We hope that you enjoy studying this Handbook and that you will successfully pass the certification exam for the IREB Certified Professional for Requirements Engineering Advanced Level Elicitation.

More information on the IREB Certified Professional for Requirements Engineering Advanced Level Elicitation can be found at: http://www.ireb.org.

Dominik Häußer

Kim Lauenroth

Hans van Loenhoud

Anja Schwarz

Patrick Steiger

Version History

Version	Date	Comment	Author
1.0.0	April 10, 2019	Initial Version	Dominik Häußer, Kim Lauenroth, Hans van Loenhoud, Anja Schwarz, Patrick Steiger
1.0.1	April 15, 2019	SOPHIST logo and credits English reviewer added. Typos fixed.	Stefan Sturm
1.0.2	July, 2, 2019	Typos fixed. Size of figures adjusted	Ruth Rossi, Stefan Sturm
1.0.3	September 12, 2019	Wrong reference in chapter 3.1.3.3 [Robles2012] replaced by [RoRo2013]. Fixed broken Link in [TiSi2017].	Stefan Sturm

1. A framework for structuring and managing requirements elicitation

In recent years, IT systems have become crucial for the functioning of businesses, government, and indeed society itself. Therefore, the high quality of these systems is essential. IT professionals have learned that the quality of an IT system is primarily determined by its requirements.

From this insight a whole new IT profession has grown: Requirements Engineering. The main idea is sharing information. Requirements Engineering as a discipline is concerned with eliciting, documenting, validating, negotiating and managing all information that system developers and operators need to build, operate and maintain successful systems.

Requirements Engineering helps all involved parties to understand what kind of system is really needed. In certain contexts, Requirements Engineering is performed by a dedicated role: "Requirements Engineer". In other contexts, Requirements Engineering is part of a larger role definition, for example: Systems Engineer [Walten et al.2015] or Digital Designer [Bitkom2017]. For reasons of simplicity, this handbook will use the term Requirements Engineer.

1.1 The scope of elicitation in Requirements Engineering

In accordance with the definition of Requirements Engineering as presented in [PoRu2015], the objective of requirements elicitation and conflict resolution is "knowing the relevant requirements", "achieving a consensus among the stakeholders about these requirements" and "understanding […] the stakeholders' desires and needs".

Within elicitation, it is the task of the Requirements Engineer to understand the stakeholders' desires and needs while ensuring that the requirements from all relevant requirements sources have been collected. This includes identifying these sources, understanding the nature and importance of the different types of requirements and applying appropriate techniques to elicit them. A major point in elicitation is to turn implicit demands, wishes and expectations into explicit requirements [ISO29148].

During elicitation, conflicting requirements from different sources are often encountered. These conflicts have to be resolved in order to create a single, consistent and agreed-on set that can serve as an input for the efficient development, maintenance and operation of an effective system.

This document describes *elicitation* and *conflict resolution* at an advanced level. This first chapter serves as an introduction to the subject and as a guide for its practical application. In **Chapter 2**, the **Requirements Sources** are described. Determining what sources are relevant is the starting point for every elicitation effort. **Chapter 3 Elicitation** gives an overview of techniques that can be used to elicit requirements, as well as guidance on how to use them. **Chapter 4 Conflict Resolution** deals with ways to resolve situations where requirements are conflicting with each other. The Handbook ends with **Chapter 5 Skills of the Requirements Engineer** which includes focal points for professionals who want to be active in this domain.

According to the IREB CPRE Foundation Level syllabus [IREB2017], Requirements Engineering has two other main activities: *documentation* and *management*. Documentation concerns ways to capture the results of the elicitation process as a means for further communication. A separate Handbook of Requirements Modeling According to the IREB Standard [CHQW2016] provides more information on one part of this topic. Management is about maintaining a set of collected, documented and consolidated requirements in a good state throughout their life cycle. This activity will be described in more detail in the Handbook of Requirements Management According to the IREB Standard.

1.2 Factors relevant to the approach of planning elicitation

Literature on software project estimation [McConnell2006] and results from industrial practice place a high responsibility for meeting overall project expectations on the discipline of Requirements Engineering. From the perspective of Requirements Engineering, a significant part of this responsibility has to be placed on requirements elicitation. This requires a specific planning approach for the following reasons:

1. Requirements elicitation cannot be planned solely based on the expected size of the outcome. It is not possible to state that we want to elicit 107 requirements and will need an average of 1.25 hours for each requirement. The reason for this is simple: we do not know the size and shape of the elicitation results. We have to elicit the requirements because we do not know them.
2. Although requirements conflicts in a project cannot be planned or predicted, in every project there will be such conflicts. Once a requirements conflict occurs and is detected, the Requirements Engineer has to react to the conflict.

Both could lead to the misunderstanding that it is not possible to schedule and control requirements elicitation by means of project management techniques. It is true that it is not always advisable to define a detailed, upfront plan for requirements elicitation (including the selected elicitation techniques, a detailed budget and time schedule). The emphasis lies on the keyword “upfront”, because a detailed, upfront plan must rely on assumptions (remember, we elicit because we do not know everything) and these assumptions are often invalidated shortly after the project start. A detailed plan is therefore only advisable if there is sufficient upfront knowledge (e.g. about the structure of the intended system) or sufficient confidence in the underlying assumptions (e.g. which aspects of the new system will be important).

The planning and execution of elicitation activities is very similar to the planning and execution of a research project. A research project typically starts with one or more research questions (or problem definitions) and defines a sequence of activities to answer and detail the defined research questions. The similarity between an elicitation activity and a research project is that the beginning of both activities is characterized by uncertainties and assumptions (or a hypothesis). The research project therefore cannot be planned completely from the beginning to the end. Instead, a research project defines activities that address selected research questions, to clarify uncertainties, or to validate (or falsify) the assumptions (or hypothesis) in the course of the project. This means in particular that the research plan and the research questions have to be reviewed, refined and updated continuously, based on the new findings.

The same applies for the elicitation of requirements. Elicitation activities usually follow an exploratory approach. At the beginning, the elicitation plan sets certain objectives (“research questions” to be answered), including a coarse timeframe and high-level exit criteria. In a number of consecutive iterations, answers to these questions are found and refined and next steps are taken based on these answers until the stakeholders accept the resulting requirements as appropriate. Modern software development processes and agile methodologies support an iterative approach consisting of short cycles in which variant solutions are produced and feedback is incorporated.

Recommended Readings

[Beveridge 1957] provides a good introduction into the definition of research projects. The full text of this book is available on www.archive.org.

1.3 Planning and executing requirements elicitation

As we have seen above, requirements elicitation requires a specific planning approach. In this section, we describe a framework for planning and executing elicitation activities. The main purpose of this framework is didactic: to support teaching and training. The framework makes the steps and information that are necessary for planning and executing elicitation explicit. The framework should not be understood as a process toolkit that can be applied directly; as with other process frameworks, practical application requires a tailoring of the framework to the situation at hand.

Finally, the planning and execution of an elicitation effort cannot be treated in isolation from other activities in a system development project. The tailoring of the approach for a particular project requires a deep understanding of the project context, taking into consideration artefacts such as the product vision statement, the project brief and the business case, and is not therefore considered in this handbook. The following discussion focuses instead on the core concepts of elicitation.

For the definition of our framework, we assume that every project that includes elicitation activities uses some kind of plan to structure the approach or tasks. This may be a sophisticated project plan including milestones, or an agile backlog.

We define two activities that can be included in any kind of plan:

- Elicitation activities to identify the requirements sources and to capture their requirements
- Resolution activities to resolve any occurring requirement conflicts

Both activities may additionally provide project management information concerning timing and resources. Details on the definition of these elicitation activities is provided in the following subsections.

1.3.1 Elicitation activity

An elicitation activity is used to plan the elicitation of requirements or the identification of requirements sources. The content of an elicitation activity is described by five elements: elicitation objective, result quality, requirements source, elicitation technique and project management information.

We use the example of an elicitation activity in a project where service engineers of a worldwide ship engine service company shall be supported with mobile devices:

ID	**RS_EA_13**
Elicitation objective	Determine the diversity (in any relevant aspect) among the user group “Service Engineer”
Result quality	Valid personas (one if they are homogenous, several if they are heterogeneous) for the user group “Service Engineer”
Requirements Source(s)	Service engineers in 5 selected service stations around the globe: Hamburg, Cape Town, Buenos Aires, Dubai, Osaka
Elicitation technique	Contextual inquiry

Figure 1: Example of an elicitation activity (ID is the project management information)

These five elements will be described in the following subsection. These elements have strong relationships with each other. Understanding these relationships helps in defining good elicitation activities. The description of the relationships is given at the end of this subsection.

1.3.1.1 Elicitation objective

The *elicitation objective* is the central concept of the planning approach since it guides all further elements. The same elicitation objective may occur several times in an elicitation project plan, for example if we want to answer a given question with different techniques or by analyzing multiple requirements sources.

The elicitation objective should be phrased as precisely as possible. It serves the following purposes:

1. It characterizes what we want to learn or understand with this particular elicitation activity, i.e. the requirements sources to be identified or the requirements to be elicited.
2. It supports the identification of appropriate requirements sources (see Section 2.2).
3. It supports the selection of an elicitation technique.
4. It can be used to measure the success of the activity (has the objective been achieved at the end of the activity?).
5. The result quality of the elicitation objective is an important indicator of the level of understanding of the system to be developed (see Section 1.3.2.2).

There are several different ways of formulating elicitation objectives, for example:

1. Formulate real questions about the requirements for the system[1].

   Example: What are the main steps of the business process that must be supported by the new CRM (customer relationship management) system and what does this support look like?

2. Formulate elicitation objectives as hypotheses that you will confirm or reject.

   Example: The smartphone app of the CRM system must provide functionalities for adding new customers. It is not sufficient to have this functionality only in the desktop part.

3. Formulate elicitation objectives based on the Kano Model, using the categories dissatisfiers, satisfiers, and delighters (see [IREB2017]).

   Example: Understand the basic factors of the existing CRM system.

4. Utilize standards or templates for specifications in your domain to define elicitation objectives.

   Example: Identify quality requirements according to [ISO25010] such as usability, reliability or security requirements.

1.3.1.2 Result quality

The *result quality* describes the intended quality of the outcome of the activity in terms of level of certainty, completeness and agreement.

[1] [Miller2009] provides a long list of questions related to non-functional requirements. [Withall2007] provides a list of requirements patterns that can be used to derive questions.

Certainty refers to the degree of evidence that can be given for the correctness of the outcome (i.e. requirements sources or requirements).

With completeness, we mean the coverage of the outcome with respect to the "theoretically possible" amount of information that could have been elicited on the desired level of detail.

Agreement refers to the degree to which the result has to be agreed on by the stakeholders. It also covers which stakeholders have to agree. Keep in mind that achieving agreement among stakeholders may require a lot of communication and therefore may create significant effort.

The Requirements Engineer should always look for a balance between the cost of additional elicitation activities and the benefits for the current project. The right level of certainty and completeness arises from a continuous interaction between the RE and relevant stakeholders.

1.3.1.3 Requirements sources

The *source* characterizes the source or sources from which the requirements shall be elicited, or the sources that are used to identify other requirements sources. We stipulate an elicitation activity focus on exactly one type of source.

1.3.1.4 Elicitation technique

The *elicitation technique* is the particular technique used to elicit the requirements from the source.

Relevant information for an elicitation technique includes:

- Estimated effort for preparation/execution/post-processing: An elicitation technique may require a significant effort in preparation, execution and post-processing for the Requirements Engineer and/or the stakeholders. For example, an interview requires preparation of the questions. Apprenticing may take several days for both the Requirements Engineer and the affected stakeholders, depending on the complexity of the project. A workshop may require analysis of the workshop results. This effort should be estimated and documented to improve the planning of elicitation techniques.
- Effort spent for preparation/execution/post-processing: In addition to the estimated effort, the real effort spent for an elicitation technique should be documented. Significant deviation between estimated and spent effort should be analyzed to understand the reasons for the deviation.
- Schedule for preparation/execution/post-processing: Besides the necessary effort, the schedule for the elicitation activity is important for Requirements Engineers and stakeholders, especially for activities that require a significant amount of effort. The schedule may also provide a deadline.
- Reference to preparation material: If a technique requires preparation material (e.g. a workshop that discusses a UI mockup), the preparation material should be referenced.

The job here is to select the optimal technique(s) for the given elicitation objective. Every elicitation technique has its advantages and disadvantages. As with requirements sources, you should not rely on one single elicitation technique. Whereas interviews may help you get detailed and specific input from talkative stakeholders, they will probably not help you with introverted people or someone who is afraid to give the "wrong" answer.

Section 3.4 provides guidance on how to reach a suitable selection of elicitation techniques.

Remember: define separate elicitation activities for each technique!

1.3.1.5 Project management information

An elicitation activity may be characterized by several project management attributes: the exact set of attributes depends on the particular project context and method.

The following list provides useful examples of project management attributes:

- Author: The person who has defined the particular elicitation activity
- Responsible Requirements Engineer(s): The project member(s) responsible for the execution of the elicitation activity
- Priority: Documents the importance of an elicitation activity for the overall project
- Dependencies to other elicitation activities: An elicitation activity may depend on the outcome of other elicitation activities. For example, a requirements workshop may need the input from stakeholder interviews.
- References to the documented requirements: Once an activity has been finished, the resulting requirements can contain references to maintain traceability between the requirements and the elicitation activity.

1.3.1.6 Utilize the relationships between the elements of an elicitation activity

The elicitation technique is the tool for performing the elicitation. The selection of the proper technique is key for the success of an elicitation activity. Nevertheless, the decision for a particular technique should be the last step in the definition of an elicitation activity.

The main reason for this is that every aspect of an elicitation activity has a relationship to the other aspects that can be used to validate and improve the overall elicitation activity. Figure 2 shows the four aspects and the six relationships among them:

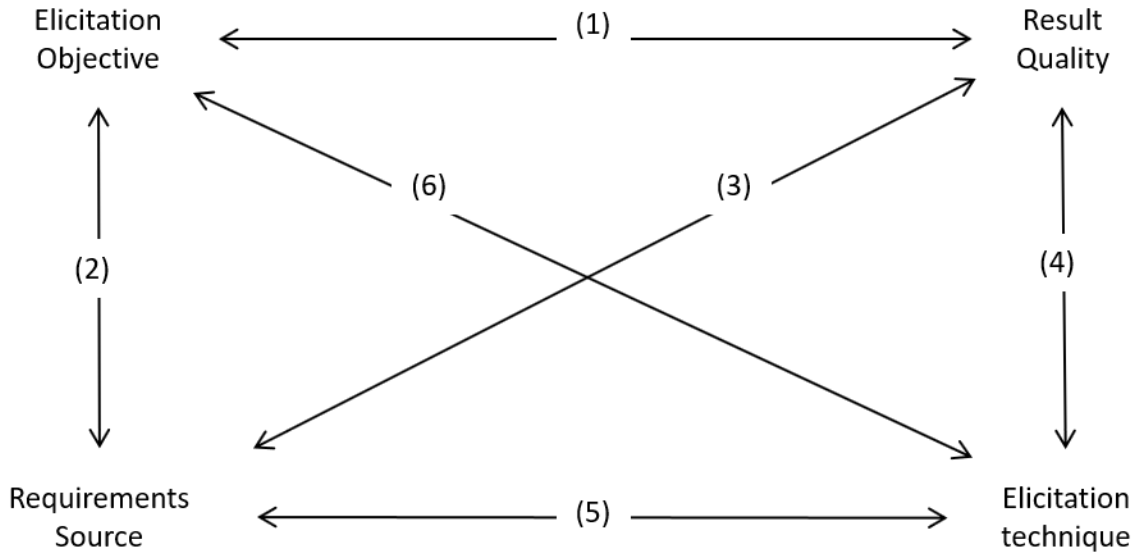

Figure 2: Relationships between elements of elicitation activities

1. Elicitation objective – Result quality: The elicitation objective must be defined in a way that it is possible to give a precise (enough) definition for the desired certainty and completeness of the result.

2. Elicitation objective – Requirements source: Is the selected source useful to achieve the elicitation objective? (An old process document may not be suitable for understanding the real business process of a company.)
3. Result quality – Requirements source: Can the selected source deliver the desired result quality? (Is it possible to clarify certain requirements from only one stakeholder?)
4. Result quality – Elicitation technique: Can the selected technique deliver the desired quality? (Is it possible to clarify certain requirements in a brainstorming session?)
5. Requirements source – Elicitation technique: Technique and Source must be compatible (it makes no sense to interview a document!), and the sources must be addressable by the technique. (Is it possible to bring together all six vice presidents of a company in a five-day workshop?)
6. Elicitation objective – Elicitation technique: Is the selected technique suitable to achieve the defined objective? (Is it possible to understand a particular business process with a creativity technique?)

These six relationships show that a suitable elicitation technique can only be selected if the three other aspects are well defined.

1.3.2 Conflict resolution activity

This kind of activity is used to resolve conflicts between requirements. Requirements are considered as conflicting if they cannot be implemented in the same system at the same time. See Chapter 4 for details.

The content of a resolution activity is described by four elements: description of involved requirements, involved requirements' sources, resolution technique and achieved result (after the conflict has been resolved).

The conflict resolution activity is described as soon as the conflict has been identified. Usually, at this time, not all aspects of the conflict resolution activity can be described. During the resolution, the remaining aspects are documented.

ID	**RS_CRA_3**
Involved requirements	RS_REQ_37 and RS_REQ_221 are not compatible. RS_REQ_37 demands wireless connection whereas REQ_221 demands wired connection. Type of conflict: not clear, yet
Involved requirements sources	RS_REQ_37 was contributed by Ms. Highmore, RS_REQ_221 originates from system archaeology of legacy system.
Resolution technique	Preferred technique: agreement Plan for resolution: Meet with Ms. Highmore and Mr. Strong (system owner of legacy system) and find a suitable solution.
Achieved resolution result	TBD (will be documented when solution has been found)

Figure 3: Example of a conflict resolution activity (ID is the project management information) in the state of identification

1.3.2.1 Description of involved requirements

The description names the conflicting requirements and specifies why these requirements are conflicting. It should describe the necessity for resolution clearly and in adequate detail, including a reference to the affected requirements. The description should further mention the type of conflict (see Section 4.2).

1.3.2.2 Involved requirements sources

The involved requirements sources (e.g. stakeholders) are the participants who have to be included in the resolution process.

For example, a requirement related to the monitoring of user activity for security reasons is in conflict with a personal data protection requirement. A security expert mentioned the monitoring requirement and the data protection requirement originates from a domain-specific law. The security expert is a stakeholder in this conflict, and the stakeholder for the domain-specific law could be the data protection officer of the company.

The identification of the involved stakeholders is an important part of conflict identification (see Section 4.1).

1.3.2.3 Resolution technique

The resolution approach defines the intended approach to resolve the conflict. The resolution approach typically consists of a description of the selected resolution technique and additional preparation activities.

Details on the selection of the proper resolution approach are presented in Section 4.3.

Achieved resolution result

After resolution, a short description of the achieved result is documented as the last step of the resolution activity. This short description is intended as a summary of the conflict resolution for people who review these activities in a later stage of the project.

1.3.2.4 Project management information

A resolution activity may be characterized by several project management attributes. The exact set of information depends on the particular project context. The following list provides useful project management information items:

- Author: The person who has defined the particular resolution activity
- Responsible Requirements Engineer(s): The project member(s) responsible for the execution of the resolution activity
- Priority: Documents the importance of a resolution activity for the overall project
- Reference to detailed conflict documentation: If a conflict relates to complex issues, further documentation should be referenced that explains the details behind the conflict.
- Reference to preparation material: If the conflict resolution requires preparation material (e.g. a detailed description for the stakeholders), this material should be referenced.
- Dependencies on other activities: A resolution activity may depend on the outcome of other conflict resolution or elicitation activities. For example, a conflict resolution proposal may require input from stakeholder interviews.

- Estimated effort for preparation/execution/post-processing: A resolution technique may require significant effort for preparation, execution or post-processing for the Requirements Engineer and/or the stakeholders.
- Latest point in time when the conflict has to be resolved: It may not be necessary to resolve every conflict immediately after its identification. If it can be postponed, the latest point until which that resolution can be responsibility left should be documented.
- Schedule for preparation/execution/post-processing: Besides the effort, the schedule for a resolution activity is important for Requirements Engineers and stakeholders, especially for activities that require significant effort.

1.3.3 Guidelines for the elicitation part of a project

In the previous subsection we defined elicitation activities; in this subsection we present guidelines for the application of elicitation activities, from planning through to execution. These guidelines are independent from any particular project management approach (for example Waterfall or Agile). The general term project may apply, for example, to a pure RE project, a business analysis project or a software project.

Hint 1.3.1:

If you are using a Kanban board (either physical or digital) for task management, elicitation activities make perfect task board items. The same also applies to Gantt charts.

1.3.3.1 Distinguishing different sets of elicitation activities

We recommend distinguishing amongst three different sets of elicitation activities:

- *Set 1 – Executed elicitation activities*: This set contains all elicitation activities that have been executed during the project so far. It describes the history of the elicitation perspective of your project and serves as a project memory. At the beginning, this set will of course be empty.
- *Set 2 – Short Term elicitation activities*: This set contains all elicitation activities that are planned for execution in the near future. The elicitation activities in this set have to be planned in detail and should be scheduled and prepared for execution. You can consider this set as the to-do list for the near future of your project.
- *Set 3 – Long term elicitation activities*: This set contains all elicitation activities that are considered important but are not yet planned and scheduled in detail. The reason for defining some activities as long term is that the current status of the project (especially existing knowledge and assumptions) does not always allow for detailed planning. You can consider this set as a backlog for objectives that still have to be further elaborated.

As the project progresses the set of executed activities will grow, as short-term activities are executed. Long-term activities will be detailed and become short-term activities, or will be refined by several short-term activities, or may be abandoned completely if they no longer make sense for the project.

We recommend distinguishing between the setup and execution phases of elicitation. In the following subsections, we provide guidelines for both phases.

In addition to the setup and execution phase, it is possible to add a *conclusion phase* that focuses on improving the elicitation skills of the project participants. In this phase, the version history of the elicitation plan is reviewed against the results (the elicited requirements and the resolved conflicts) to learn from successes and failures in the finished project. In iterative projects, this may take place after each iteration.

1.3.3.2 Guidelines for the setup phase

The initial set of elicitation activities is defined in the *setup phase*. This initial set describes the intended approach for the elicitation of requirements. It is based on the specific characteristics of the project (e.g. the development approach) and the existing knowledge and assumptions of project participants. Resolution activities are typically not defined in the setup phase, since requirements conflicts are unknown at this stage. However, if there are known requirements conflicts (for example from previous projects) or indicators for potential conflicts, a plan for addressing these should be incorporated as soon as possible.

The term *setup phase* does not imply that you have to spend several days or even weeks to develop the initial set of activities. Nevertheless, the setup phase is of high importance for a clean project start and for enabling effective elicitation of requirements. Remember, elicitation is similar to a research project and a good research project requires the definition of proper research goals to be effective.

Get an overview of the project situation and the business case

Every project is unique. What might have worked for one development project may be completely wrong for another project. It is important to analyze each new project to get a clear picture as to which elicitation activities are suitable. First, you need to understand the nature and the context of the project. You may be completely new to the domain and/or organization or you may have been working there for many years. If the latter, some of the issues discussed in the following section may not be less important due to your experience.

A development project is also strongly influenced by its **domain**. Do you know enough about that domain to understand how it influences the project and to know what is important? If you are new to a domain, you will have to read (at least introductory) literature and/or websites to understand the domain context and terminology. A common elicitation objective to develop an understanding of the domain is the creation of a glossary (see also [IREB2017]). You should further look for people who can help you understand the domain and ask them to explain to you what you need to know in the context of your project.

It is important to understand a project's history. No project appears out of thin air; even a brand-new project has a history. You have to know about this history in order to understand the project's objectives and to avoid pitfalls. Some questions you should ask in this context are: Why was this project initiated? Who initiated it? Have there already been failed attempts to reach the goal of the project? If yes, who was involved in those approaches? Why have those approaches failed? Has a pre-study been conducted? If yes, who was involved and what were the results (potential requirements sources!)? Understanding the history of a project will not only help you in defining the elicitation objective and identifying the right requirement sources, but also during resolution activities (identifying a conflict, understanding a conflict, knowing the roots of a conflict).

Of course, you should also talk to the project initiators, project leaders, project members and anybody who may be able to provide you with information concerning the project (for more details on stakeholder analysis, see Chapter 2). Who are the "important" people in your project? Who knows what about the project, about the technologies used, about the domain, etc.? What are the project goals? What are the timelines? How is the project organized? Who is responsible for what? These are just a few questions that will help you understand the development project and its context. You should also find out whether there are specific **project constraints** that may influence the elicitation approach.

You should also try to get a clear picture of how **complex** the project is or might become. What influences the complexity of the project (complexity drivers)? Are there any indications that the project may be (much) more complex than currently assumed? Are there any regulations concerning project organization or artefacts that have to be met (e.g. Automotive SPICE, GxP[2])? How many people or companies are involved in the project? Is the project team located at the same site or is it (globally) distributed? How about the potential stakeholders? Is it already clear that some of them are difficult to reach?

Determine elicitation objective

We recommend starting the elicitation part of a project with the definition of the elicitation objectives. The elicitation objective is the core element of the elicitation activity, since it guides what we want to learn about the requirements or their sources as well as which elicitation technique we should apply to reach that elicitation objective.

An initial list should be created first, which shows the level of your (or the team's) understanding of the project. If the project participants can agree on a list of detailed questions, this shows that they know what they want to learn in the project. Detailed elicitation objectives can be turned into short-term elicitation activities and scheduled according to project priorities.

A list of abstract or vague elicitation objectives may be an indicator for a weak understanding of the targeted project outcome. Such a situation is not uncommon. Select two or three objectives that appear to be the most important ones and plan a short-term elicitation activity to get a better understanding for these. Afterwards refine the objective. Vague elicitation objectives with high importance for the project should be detailed by one or more short-term elicitation activities. Vague elicitation objectives with less importance should typically be turned into long-term elicitation activities for consideration later in the project.

Plan for the systematic analysis of the system context

From the foundation level, we know that the system context is essential for the identification of requirements sources and for understanding requirements (see [IREB2017]).

A highly recommended elicitation objective, therefore, is **understanding the system context.** Recommended elicitation objectives for acquiring an understanding of the system context are:

- Identify people (stakeholders or groups of stakeholders) related to the system
- Identify systems in operation (other technical systems or hardware) related to the system
- Identify documents
- Identify processes (technical or physical processes, business processes) the system is involved in
- Identify events

[2] GxP is a general abbreviation for "good practice" quality guidelines and regulations. The "x" stands for the field of application, for example GAP for "Good Agricultural Practice".

Plan for the systematic and pragmatic identification of (multiple types of) requirements sources

The right requirements sources are a key asset for the success of requirements elicitation. Understanding the domain and the organization is essential for the systematic and pragmatic identification of requirements sources. Note that the precision with which a source is defined may vary for short- and long-term elicitation activities (see above):

- Requirements sources for short-term elicitation activities have to be named (e.g. name the specific stakeholders for an interview, or name the standard that shall be analyzed). This is necessary to make the elicitation activity executable.
- Requirements sources for long-term elicitation activities may also be defined as types or categories of requirements sources.

Understanding the **organization** and its culture is also important. Even within the same domain two companies may have a totally different culture and way of doing things. Depending on the organization, there may also be huge cultural differences between subsidiaries or even departments of that organization. Make sure you know the unwritten rules within the organization. This includes simple things like how people are addressed (e.g. is it usual to address somebody by their first or last name?). You should also know what role hierarchy plays in that organization. Up to what level can you invite people in leadership positions directly, and when do you have to make an appointment with their office? Apart from the organization's culture, you should also know how it is structured. Which departments are there? How are the departments connected to each other? And, of course, you should know how the organization creates cash flow (i.e. you should know what it produces/sells – even if that has nothing to do with the development project for which you are eliciting requirements!).

Consider relevant process patterns to define the activities

At first glance, the framework described in this section may appear overwhelming and complicated. This is not, of course, the intention. The framework described here is an abstraction of the best points of various approaches found in the literature.

The literature provides several sophisticated methods that support the elicitation of requirements. Popular methods are, for example, human-centered design and Design Thinking. Using our framework, these methods can be considered as a sequence of several elicitation techniques and require a significant amount of planning and effort. At the same time, the method description in the literature provides several hints for the definition of elicitation objectives and requirements sources.

We highly recommend planning more than one elicitation activity when using a method. Different activities may for example correspond to different techniques belonging to the same method.

To support you with the application of methods presented in the literature, we have developed the concept of process patterns in requirements elicitation. Section 1.4 introduces this concept as a separate topic.

Allow time and budget for resolution activities

Though there may be no conflicts in the beginning of a project, you should plan time for resolution activities. Not every conflict might be a showstopper, but there will be conflicts you have to solve:

- Allow time for actively finding conflicts, because the earlier you do so, the greater the possibility that you find time to solve them.
- Allow time for solving new conflicts before having concrete evidence of their existence.
- Schedule conflict analysis as early as possible, as much useful knowledge can be gained and it may prevent having to deal with uncooperative stakeholders later.

Go for quick wins. If you can avoid a conflict arising, or find a quick solution (even when not explicitly planned), use the chance. You might not have all stakeholders in your workshop any time soon.

1.3.3.3 Guidelines for the execution phase

The *execution phase* focuses on the execution of elicitation activities. In this phase the elicitation activities are performed according to the existing plan.

Consider elicitation activities as time-boxed activities

A serious risk in requirements elicitation is wasting time and resources on ineffective activities that do not achieve the expected results. We therefore recommend considering elicitation activities as time-boxed, where each time-box adds information iteratively and incrementally to the requirements set.

If an activity does not meet expectations for effort and/or schedule, stop the activity and examine the achieved results to understand the reason for the failure. There may be several reasons for potential failure of an elicitation technique: for example, the conflict resolution technique does not fit with the conflict situation, or the addressed requirements sources are not able to provide adequate information. The results of this examination can be used to plan new activities, or to refine existing activities.

Question the plan after each activity (and revise if necessary)

The existing elicitation plan should not be considered as fixed in stone. The plan was created with the information available at the time, including assumptions. The acquisition of new information, the occurrence of requirements conflicts, and the refutation of existing assumptions are daily business in elicitation.

Adjust your plan continuously, using knowledge obtained about the organization, stakeholders and project complexity during ongoing activities. Do not focus only on elicited requirements, but also on all other information. The following questions may serve as a checklist to review your plan after each elicitation activity:

- Do the results of an activity have an impact on the defined short-term activities?
- Are the objectives of the related activity still valid?
- Can the results of an activity be used to refine existing vague or abstract objectives?
- Is it possible to revise existing long-term objectives?
- Do the results of an activity lead to new objectives?
- Do the results indicate a new requirements conflict?
- Are the results useful in resolving any existing requirements conflict?

Schedule defensively and make use of short- and long-term elicitation activities

Besides the precise definition of elicitation activities, the proper execution sequence is an important factor for successful requirements elicitation. Especially in project situations with a high level of uncertainty, we recommend a defensive approach to the scheduling of elicitation activities to avoid a waste of resources.

The scheduling of elicitation activities is driven by the following factors:

- Availability of stakeholders/requirements sources: An activity can be performed only if the necessary stakeholders (for elicitation activities) or requirements sources (for elicitation activities) are available.
- Availability of Requirements Engineers: An activity can be performed only if skilled Requirements Engineers are available to execute the activity.
- Value to the project: Activities with a high value to the project should be executed with priority. The definition of value depends on the particular situation. Some value definitions are, for example:
    - Importance of the requirements that an elicitation activity delivers (defined by the elicitation objective)
    - Importance of the conflict to be resolved
    - Importance of the delivered information for further elicitation activities (e.g. an elicitation activity has been defined to detail a vague but important elicitation objective)

Incorporate slack to leave time for creativity, conflict resolution and unexpected events

Be aware of the fact that you cannot plan everything upfront, as you have to deal with a high level of uncertainty. Do not be surprised if on your way urgent new activities pop up. If you did not foresee some slack to deal with it, your other, planned activities will suffer and there will be no room for creativity. In an early phase of an elicitation project, 25% of slack is not uncommon. Later on, this percentage can be reduced, but will never be zero.

Parallelize independent activities

Parallelizing independent activities can increase the efficiency of elicitation activities. Two or more elicitation activities are considered independent from each other if the elicitation objectives are independent from each other and the requirements sources are not identical. It is not advisable to parallelize activities that have dependent (or even the same) elicitation objectives, because the result of one activity may have an impact on the other.

Two or more resolution activities are considered independent from each other if the conflicting requirements are independent from each other and the conflict participants are different. It is not advisable to parallelize activities that have interdependent conflicts, because the resolution of one conflict may have an impact on the other.

Make sure, however, to coordinate regularly among all parallelized activities. Otherwise, you might miss hidden dependencies or findings that might help with the other activities.

Combine elicitation activities that address the same requirements source

Elicitation activities that address the same requirements source (e.g. the same stakeholder or stakeholder group) can be combined to increase the efficiency.

For example, in the case that we want to address three different elicitation objectives with interviews and the stakeholder for these objectives is identical, it is possible to schedule one interview and work on all three objectives in that meeting.

Search for conflicts and react to them according to an agreed strategy

Actively searching for conflicts is a daily task. No matter whether you perform elicitation activities, specification reviews, quality assurance on a requirements model or any other task, you should develop good skills in checking for consistency at all times.

Document potential conflict information, for example conflict indicators in your stakeholder list, for later review. For social or mixed conflicts, be aware of the indicators. Not every indicator may lead to a real conflict, but every conflict has indicators. It is always useful to know where to observe before it is too late to react.

Always stay neutral in a conflict and try to find a sustainable solution for the whole project and all its stakeholders. It is also useful to be perceived as neutral when taking the role as moderator in the conflict situation or workshop.

Examples on how to find conflicts in the execution phase are:

- Look for inconsistent terms in your glossary or terminology model.
- Count: If your GUI has five fields and you defined six labels, there must be a mistake!
- If stakeholder A wants a red button, the same button cannot be green for stakeholder B.
- If you defined a behavior for a condition, did you define an alternative behavior when the condition does not apply as well?

1.4 Process patterns

We know from industry practice that every project is a unique event. It is therefore a challenge to provide concrete guidance for performing requirements elicitation, since there are numerous factors that may influence the best approach. Some of these factors are available time and budget, type of system to be developed, availability of stakeholders and the experience of Requirements Engineers.

Nevertheless, there are certain methods from the literature and industry practice that have proven their usefulness in various situations. In addition to describing some of these methods, we introduce the idea of process patterns. The concept of patterns was originally developed in an architectural context [AlIS1977] to document reusable knowledge for creating architectures, and was later transferred to software development with the introduction of design patterns for software [GHJV1994].

In an elicitation context, process patterns provide a form for documenting useful and proven ways for performing requirements elicitation. The main objective of this section is to teach the underlying idea of process patterns as a toolkit instead of teaching specific approaches. First, the structure of process patterns for requirements elicitation is described. Afterwards some examples for process patterns are given.

1.4.1 Structure and benefits of process patterns for requirements elicitation

A process pattern consists of the following elements:

- *Scope*: This section describes the project situation or situations that are suitable for the application of a pattern. This description may also include particular situations in which the application of the pattern is not advisable.
- *Necessary Effort/Resources*: This section describes the effort and resources that are necessary to apply a certain pattern. The effort is typically described in terms of time and personnel resources. Additional resources may include, for example, workshop material, special software or special locations.
- *Pattern elements*: This section explains the details of this pattern. The description includes references to methods applied and concrete descriptions of elicitation activities so that the reader can understand the pattern in terms of requirements elicitation.

- *Instantiation*: This section provides specific details on how to get started. The description includes activities for preparing the pattern and an initial plan with elicitation activities (see Section 1.3.1.1) that can be used as a guideline for starting.
- *Further reading*: This section provides additional references to useful literature with further details on the pattern.

Depending on the origin of the underlying ideas of the pattern, the details described are not necessarily restricted to requirements elicitation, and may include other activities such as design or testing. Such non-elicitation related parts should be kept short. Where possible, the description will describe the outputs of such non-elicitation activities for subsequent use in requirements elicitation.

The Requirements Engineer should actively search for patterns that are relevant for his or her own situation. Mind that:

- Process patterns are good practices from literature and practice, providing a starting point for defining elicitation activities in comparable situations.
- Process patterns provide a meta-perspective on the way of working. Instead of applying the same approach in all project situations, process patterns allow the development of a toolbox beyond specific techniques (e.g. interviews).
- Typically, the information provided is not sufficient for an immediate execution of the process. Analysis of similarities and differences between the pattern scope and the actual situation helps in identifying a suitable approach and in selecting appropriate techniques.
- The list of patterns covered in this chapter is neither complete nor exhaustive. Furthermore, patterns can, and often should, be combined in various ways.
- Experienced Requirements Engineers are encouraged to develop and share their own patterns.

This section presents three example patterns: Waterfall, human-centered design, and Design Thinking. Waterfall is selected as a pattern because the Waterfall process model is often confused with Requirements Engineering as a method. The description here should make clear that Waterfall is only one way for performing Requirements Engineering / elicitation.

Human-centered design and Design Thinking are presented because they are two very popular approaches for eliciting requirements and combine various techniques (e.g. workshops and creativity techniques) with a process model that allows for an easy application in various situations.

Agile development has intentionally not been included here as a pattern because the details on Requirements Engineering in agile projects are covered by the IREB advanced level module RE@Agile.

1.4.2 Waterfall

The Waterfall model describes a linear-sequential software development life cycle model, in which the development of a system proceeds through a sequence of different phases. Royce [Royce70] first described this model with the phases Requirements, Analysis, Design, Coding, Testing and Operations. Other phase definitions have been published, e.g., the DOD-STD-2167A standard of the United States Department of Defense [DoD88]. The V-model [FoMo91] or Boehm's spiral model [Boehm88] are variations of the Waterfall life cycle.

1.4.2.1 Scope

This type of development is mostly seen in large, long-lasting projects, developing technical (embedded) systems, complex systems with interfaces to other (legacy) systems and following a big-bang strategy for implementation.

Most obvious characteristics of this kind of projects are:

- A clearly defined business goal;
- Precise understanding of the system to be developed;
- Precise understanding of the domain;
- Precise understanding of technology that is intended to be used;
- Strong directives from a single business owner or steering committee;
- A strict approach, defined in a formal plan with budgets and timelines right from the start, and controlled by strong project management;
- Teams of different professionals working on the individual phases;
- Outputs of each phase serving as inputs for the next phase, passing information through formal documentation;
- Quality gates between phases, with formal management decisions to proceed;
- After passing a quality gate, the resulting requirements are “frozen” as specifications and, if subsequent changes are required, they follow a formal change procedure.

1.4.2.2 Effort / resources

Waterfall projects usually have a long duration (months through years) and require a large effort (often several hundred man months). At the start of the project, a detailed plan is set up and agreed for the rest of the project. This plan includes an initial requirements phase that may last from some weeks to several months, and may engage several professionals.

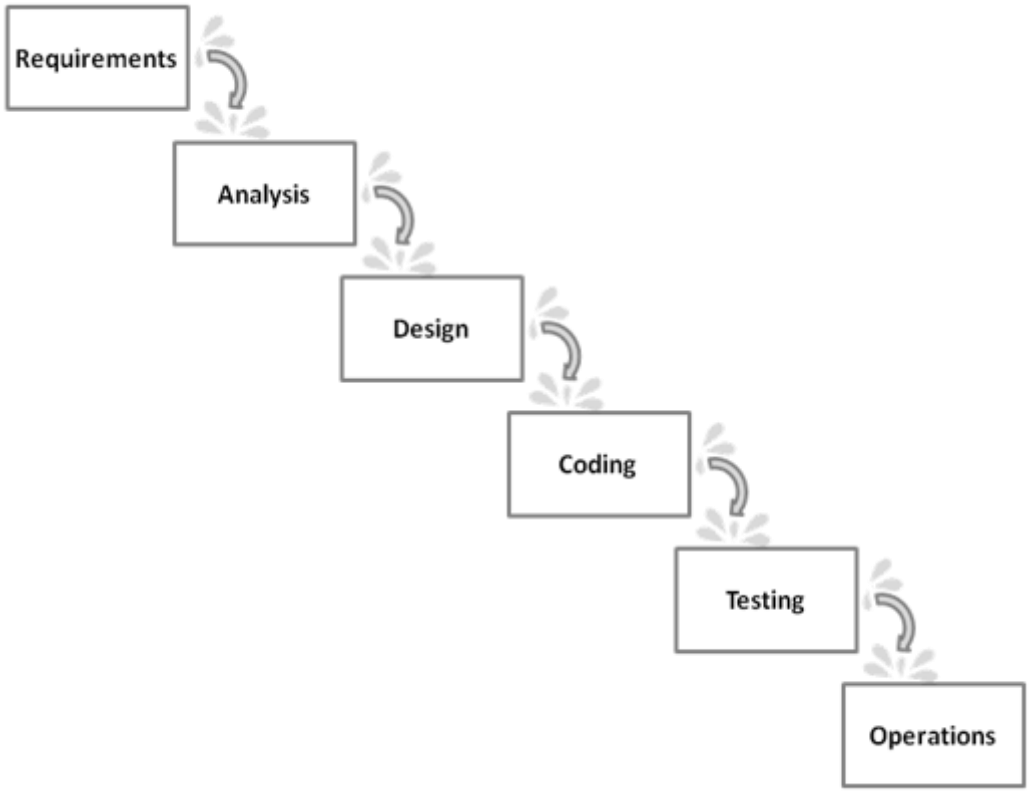

Figure 4: Waterfall process

1.4.2.3 Pattern elements

There are different phase definitions for the Waterfall model, usually variants of Royce's original definition (see Figure 4). From a Requirements Engineering point of view, the requirements phase may be considered the most important. In this phase, the main requirements are determined, setting the overall direction for the rest of the project. Although less prominent, Requirements Engineering continues to play a role in all subsequent phases.

The following paragraphs describe each phase in more detail including guidelines for elicitation activities.

Requirements

The goal of this phase is to translate the high-level business goal formulated by the business owner into a validated set of major (business) requirements. In addition, the main constraints that define the solution space for the system and the project are identified.

Requirements Engineers typically elicit the requirements by having qualitative interviews with a (usually small) group of executives and managers, summarizing the results in reports and processing their feedback. It may be difficult to get enough time from the interviewees, due to their overcrowded agendas. Apart from interviews, clarity about the requirements can also be achieved through document analysis.

The results are consolidated in a detailed end report that passes through a series of draft versions, which are reviewed repeatedly. Conflicts are solved by processing feedback until a consensus is reached, or by overruling through higher management. At the end of the phase, a formal inspection on the latest draft serves as a quality gate, after which the end report is released for input to the next phase.

Analysis

The analysis phase focuses on elaborating the information architecture, the user interface and interfaces with other systems into a set of system requirements.

A major task is to elicit the constraints arising from the surrounding IT landscape, mainly through document analysis and interviews with system administrators and other IT professionals. On the user side, the focus shifts to the direct end users. Both interviews and questionnaires can be used for elicitation, depending on the size and complexity of the user community. Workshops with user groups may also be relevant. If the user community is diverse, personas may be useful. A pitfall might be to "forget" external users and other indirectly affected clients, assuming that the internal users represent their opinion sufficiently.

Once again, this phase results in a consolidated and validated end report, that passes through a quality gate before release.

Design

In the design phase, the system requirements are elaborated into software requirements as a blueprint for the system to be built.

In this phase, IT professionals are in the lead: functional and technical designers, database and system administrators, technical operators. Interfaces are defined in more detail, and feedback from user representatives is gathered and processed. Low-fidelity prototypes can be useful. Observational techniques may be applied to align the designed system with the operational processes of the end users. An important task may be to reconcile quality requirements with technical constraints.

At the end of this phase, the complete set of business, system and software requirements will have grown to a large and complex collection that must be actively managed (change and configuration management, baselines, versioning, prioritization, traceability). Maintaining consistency between all requirements is a major concern.

The design phase results in one or more design documents, which will also pass through a quality gate before release. Where outsourcing is used, formal design documents form the basis of the contract with the outsourcing party. In such cases, the quality of the requirements in the design documents is crucial for project success.

Coding

The purpose of this phase is to build a technical solution that fulfils the requirements established during the previous phases. IT developers like technical designers, database specialists and programmers are involved.

In theory, no new requirements are developed. In practice, developers may detect inconsistencies between requirements, conflicts with overlooked low-level constraints, or misinterpretations and miscalculations from previous phases. Sometimes, high-level prototypes are used early in this phase to obtain feedback from the end users.

Component and integration testing serve as quality gates for the next phase.

Testing

In the testing phase, independent testers, end users, system administrators and operators try to verify whether the system meets the defined requirements and to validate if the system will support the intended business processes without major risk. Once again, no new requirements are expected, but the resolution of defects may require reconsideration and adjustment of previous solutions.

The testing phase is itself the final quality gate, providing the information needed by senior management to make a go-live decision.

Operations

In the operations phase the system is used in a live business situation.

Incidents in the operational system and changes in the business environment may occur, leading to new or updated requirements. Impact analysis is applied to decide on requests for change. It is important to incorporate all accepted changes and to make sure that a single, consistent, up-to-date set of requirements is maintained throughout the lifetime of the system.

1.4.2.4 Instantiation of this pattern

In the Waterfall pattern, the demand for result quality is very high. Low quality in early phases inevitably leads to projects with time and cost overruns and systems that do not meet customer expectations.

The quality gates at the end of each phase serve as a guard against low quality. In the early phases, reviews and inspections are the main techniques to keep the quality at a high level. It is good practice to engage participants not only from the current phase, but also from the previous and the subsequent phases. After passing the quality gate, the requirements are "frozen" as specifications for the following phases. This means that if someone wants to change a requirement later on, a formal change procedure must be followed, in which the impact of the change is analyzed and evaluated. In practice, this often means that changes are discouraged. In this way, specifications from earlier phases act as constraints for the next ones. This is both a strength – it brings stability – and a weakness – it causes inflexibility – of the Waterfall pattern. Therefore, this model is best used in stable business environments where a clear goal must be realized in a highly controlled and auditable way.

When starting Requirements Engineering in a Waterfall project, you will concentrate on the requirements phase. Though seemingly contrary to the nature of such a project, consider an iterative, time-boxed approach, in which the leading business goal is refined stepwise into a set of business requirements. The time-boxes should fit within the overall project planning to yield an end report in time for the start of the analysis phase.

The following table shows an example list of objectives.

No.	Elicitation Objective	Elicitation Technique	Requirements Source	Result Quality
1	Identify relevant stakeholders and documentation	Interview	Business owner	Medium
2	Prepare for interviews	Document analysis	Relevant domain and IT documents	Medium
3	Understand business needs	Interview	Relevant executives	Medium
4	Obtain feedback	Distribution for comments	Relevant executives	Higher
Repeat steps 2 – 4 until feedback proves that the quality level is satisfactory				
5	Consolidate overall picture	Workshop	Business owner, interviewees, other stakeholders	High
6	Validate results	Inspection	Relevant stakeholders and representatives from next phase	High
7	Obtain acceptance	End report	Business owner and relevant stakeholders	High

In the next phases, the Requirements Engineer may not be in a leading role. Detailed system and software requirements will be developed and refined together with other professionals. The main concern of the Requirements Engineer will be the management of the growing set of requirements. The Requirements Engineer often acts as consultant and coach on requirements issues, solving conflicts and guarding consistency.

Make sure that sufficient Requirements Engineering capacity is available during these phases, in particular during (acceptance) testing, where Requirements Engineers should take an active role in ensuring that the system actually behaves as specified in the requirements set.

1.4.2.5 Further reading

[Royce70] W. Royce: Managing the Development of Large Software Systems. In Proceedings of IEEE WESCON 26 (August): 1–9, 1970

[DoD88] United States Department of Defense: DOD-STD-2167A, MILITARY STANDARD: DEFENSE SYSTEM SOFTWARE DEVELOPMENT, 1988

[FoMo91] K. Forsberg, H. Mooz: The Relationship of System Engineering to the Project Cycle. In: Proceedings of the First Annual Symposium of National Council on System Engineering, October 1991: 57–65.

[Boehm88] B.W. Boehm: A Spiral Model of Software Development. In Tutorial: Software Engineering Project Management. Edited by R.H. Thayer, IEEE Computer Society Press, Washington D.C., 1988, pp. 128–142.

1.4.3 Human-centered design

Human-centered design (HCD), also known as user-driven development (UDD), is a framework of processes (not restricted to interfaces or technologies) in which the context of use, usability goals, user characteristics, environment, tasks and workflow of a product, service or process are given extensive attention at each stage of the design process [RiFl2014].

The usability of a system is the extent to which the system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use [ISO9241.11].

Effectiveness is the accuracy and completeness with which users achieve their specified goals.

Efficiency means the resources expended in relation to the accuracy and completeness with which those goals are achieved.

Satisfaction refers to freedom from discomfort and positive attitudes towards the use of the product.

Given the aforementioned definition, it is clear that "usability" is not something that can simply be mentioned in an interview and then defined as a non-functional requirement, e.g. "the system must be usable". The usability of a system depends on the tasks of its users and the context of use. Human-centered design is an approach for identifying these interdependent elements:

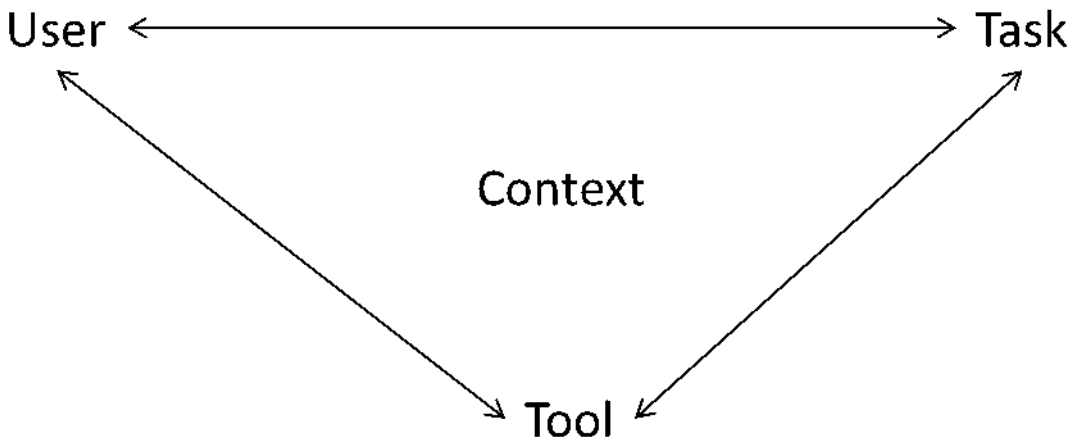

Figure 5: Mutual depending elements that need to be considered in HCD [Shackel1991]

User experience (UX) refers to the concept of multi-channel contacts between a customer and a supplier. The customer experiences a journey from his or her first contact with the supplier (probably via an advertisement or sales representative) through various interactions with online or offline touchpoints (e.g. searching for a product in the online shop, buying it, dealing with customer service and paying for it), until he or she terminates the customer relationship. Hence, the job of the supplier is not to optimize the usability of one specific channel, but rather optimizing the entire user/customer experience. In Shackel's model (see Figure 5), this is one of the important aspects of a tool's context: the tool is just one of many touchpoints to the user.

1.4.3.1 Scope

The HCD pattern applies to projects in which there is a high level of interaction with users. If interactions are mostly to other systems, this might not be an appropriate pattern. However, most systems eventually serve human beings, and it is therefore worthwhile finding out who they are (user groups), what they do (tasks) and in which context will they operate the new system.

1.4.3.2 Effort / resources

If a large, new system is to be developed and the project team is new to the domain and/or freshly assembled, a significant, upfront thinking phase (up to 3 months) should be planned. This gives the team a chance to learn about the domain, to research the users, their tasks and their context, and to identify and resolve major risks. Initial development setup, e.g. establishing of development environments, continuous integration, a first version of the solution architecture, etc., should also take place. UX skills are needed, with more professionals required if there are multiple user groups to be investigated.

If the system is less complicated, the development team is well established in the domain and has a long record of successfully working together, or if the project is only an extension or maintenance fix to a system previously developed according to HCD, then the upfront thinking period can be shortened to a few weeks or even days. Fewer specialist UX skills are needed, though it is good practice for all members of the project team be aware of the importance of UX and to have a human-centered mindset.

1.4.3.3 Pattern elements

HCD can be executed as a process covering the entire system development – refer to ISO 9241 210, the Human-Centered Design Process:

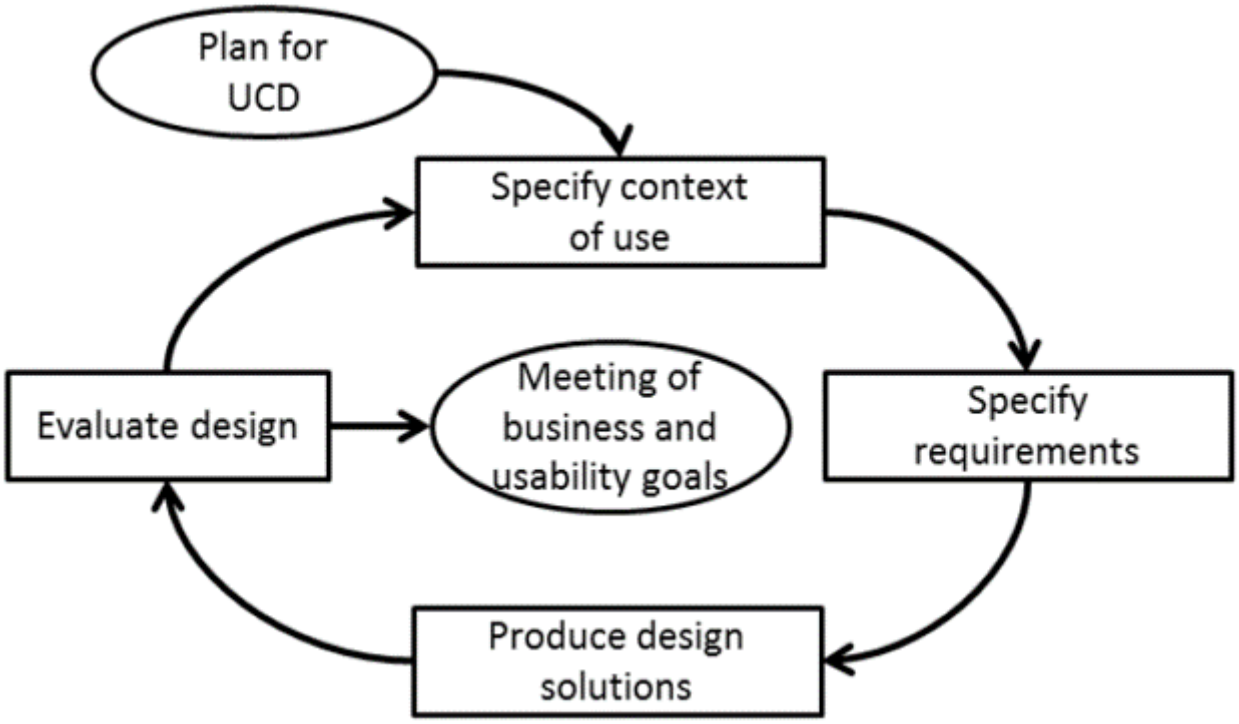

Figure 6: Iterative, human-centered design (HCD) for interactive systems [ISO9241.210]

However, HCD is an approach or even an attitude rather than a process model. Often it is integrated into engineering process models (e.g. agile approaches or Waterfall, see Section 1.4.2) or even an integral part of the design processes (e.g. Design Thinking, see Section 1.4.4) to cover aspects addressing the most crucial stakeholder - the user.

1. Plan for HCD
   Plan the necessary HCD activities and ensure they are established in the overall project plan (see also the next section "Instantiation of this pattern"): The steps are executed iteratively until the business and usability goals are met.

2. User research (specify context of use)
   First, the Requirements Engineer identifies and analyzes the users of the system to be developed. This user analysis should capture the context, characteristics, goals, attitudes and tasks of an adequate number of actual users (not their superiors!), aggregate them into user groups and document these, e.g. in the form of a persona per user group.
   Once the user groups are known, a conscious decision can be taken as to which shall be considered as the primary user group to be supported by the solution. The system, and especially its user interface, will be optimized for these users. It can then be decided if the other user groups shall be addressed via the same user interface, or whether separate user interfaces must be provided. It may even be decided that some user groups will not be addressed by the system at all.
3. Specify requirements
   The Requirements Engineer selects elicitation techniques that will help to gather information about the tasks performed by the users and the artifacts they use. Given the users, tasks and context, the requirements for the system can be specified.
4. Produce design solutions
   When validating requirements with real users, avoid complicated models and cumbersome specification documents which they don't understand. It is more effective to show prototypes or let them walk through scenarios or storyboards.
5. Evaluate design
   Apply usability evaluation in any form: for example, informal hallway testing (a short walkthrough of your prototype with a few members of the staff who happen to walk down the hall and can afford 5-10 minutes of their time) is a fast, cheap and informal technique, or on the other end of the scale are formal, lab-based usability tests with a larger number of externally recruited users.

1.4.3.4 Instantiation

No.	Elicitation Objective	Elicitation Technique	Requirements Source	Result Quality
1	To learn about the tasks of users (to prepare for the contextual inquiries)	Interview	Specific experts who know what users do (or should do): Managers, trainers, process engineers, user representatives, etc.)	High certainty / low completeness
2	To learn about the tasks of users (to prepare for the contextual inquiries)	Perspective-based reading Study training material, participate in training courses Explore the user's systems (training systems, test systems)	Training materials Training system Test systems	High certainty / low completeness
3	To identify user groups in context	Contextual inquiry (CI) Several CIs	A few specific users	Medium certainty / low completeness
4	To learn about users in public context (e.g. using an information system in a public place)	Field observation	Many casual users	Medium certainty / low completeness
5	To gather quantitative data on specific questions	Survey with Questionnaire	Many users at different places	High certainty
6	To understand a specific business	Apprenticing for a few days	One or two specific users	High certainty / low completeness
7	To validate requirements and to find new requirements	User Walkthrough based on: - Prototyping - Usage Scenarios - Storyboards	Users	High certainty / high completeness

Hint 1.4.1:

One of the first lessons in HCD for development project members is to recognize that "I am not the user". Even if the application is targeted to users just like me, other users are so different in attitude, experience, goals, tasks, etc., that there is no substitute for their direct feedback. Even a representative user joining the project team becomes after a short time "contaminated" by discussions taking place in the development team (e.g. he or she knows the concept that led to the design of a complicated screen), so that he or she is not really any more representative.

1.4.3.5 Further reading

[RiFl2014] provide a brief but very practical overview on human-centered engineering, i.e. creating products for humans.

[HaPy2012] provide a comprehensive overview on UX.

[BeHo1998] are the inventors of contextual inquiry.

1.4.4 Design Thinking

Design Thinking is a formalized process for the practical and creative resolution of problems and for creation of solutions, with the intent of an improved future result. It is a form of solution-based, or solution-focused, thinking – starting with a goal (a better future situation) instead of solving a specific problem. Several Design Thinking approaches exist (see [A4qu2018]). A widespread approach is d.school from the Hasso Plattner Institute of Design at Stanford University (see [Dsch2012]).

1.4.4.1 Scope

Design Thinking is useful for creating alternative solution ideas and possibly requirements, either for the overall project goal or for some specific aspect (e.g. certain features of the system). Design Thinking is therefore applicable at the very beginning of a project (e.g., to develop innovative ideas for a web shop) or during the project when elaborating on some particular aspect (e.g., a customer overview screen in a policy admin system). An alternative to this pattern is the Human-Centered Design pattern, also described in this handbook.

1.4.4.2 Effort / resources

Design thinking is a scalable method. A whole Design Thinking process can be performed within a few hours, days or weeks, depending on the available resources. The literature (e.g., [LiOg2011]) recommends that the Design Thinking team work in a dedicated room, as the team produces various outputs and depending on the techniques applied requires work space (e.g. for developing storyboards or other types of canvas techniques).

1.4.4.3 Pattern elements

In the d.school approach, the Design Thinking process consists of five iterative phases. The participants are project stakeholders and Requirements Engineers. One Requirements Engineer is also responsible for moderating and driving the process. Design Thinking emphasizes a multidisciplinary setup for the participants, for example: users with different ages, representatives from the business as well as from technical teams, people from other groups related to the project (e.g. NGOs). From a Requirements Engineering perspective, we recommend performing elicitation activities that systematically analyze the project context in order to identify an optimal group of stakeholders for the Design Thinking process.

Although the phases are presented sequentially, the Design Thinking process emphasizes the importance of iterations. If the results of one phase lead to a better understanding or modification of already developed results, the process has to go back to the corresponding phase and modify the results. For example, if the prototype phase created results that allow for a better definition of the project goal, the process should go back to the define phase.

In the following, we will describe each phase of the d.school approach in more detail including guidelines for defining elicitation activities.

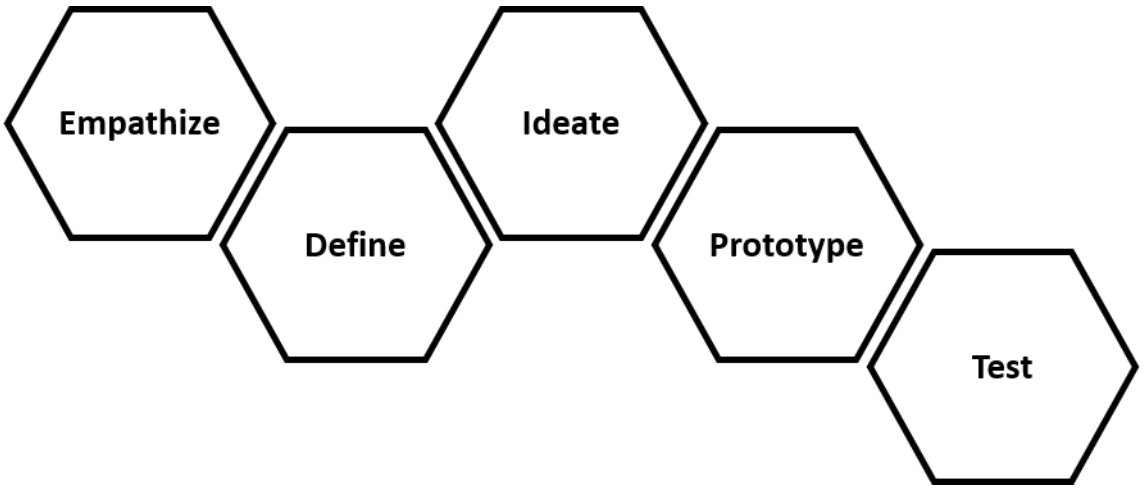

Figure 7: d.school Design Thinking approach [Dsch2012]

Empathize:

The goal of this phase is to understand the people that are impacted by the project outcome (e.g. the users of the software), to understand their way of working, their physical or emotional needs, or the environment in which they work.

From a RE perspective, this phase focuses on the definition of the system context and identification of the stakeholders (and other requirement sources). To perform this phase, you have to define elicitation activities with elicitation objectives directed towards understanding the people within your system context. The requirements sources are stakeholders that are outside the Design Thinking team. The result quality depends on the overall project goal: nevertheless, all information is welcome that supports gaining an understanding of the stakeholders. Possible techniques for this phase are qualitative interviews (see Section 3.1.1.1), requirements workshops (see Section 3.1.1.3) and observation techniques (see Section 0). Artefact-based techniques (see Section 3.1.3) are also useful in case you need a better understanding of the domain or the technology. An appropriate way of documenting your findings would be personas (see Section 2.2.4)

Define:

Process and synthesize the findings from your empathy work in order to form a user point of view that you will address with your design.

From a Requirements Engineering perspective, this phase is directed towards the documentation of achieved results. An important outcome is a concrete definition of the project goal within your Design Thinking team. You can define elicitation activities with elicitation objectives that are directed towards the understanding of relationships and dependencies. The result quality has to be very high since the results have to reflect the common understanding of your project team. The requirements sources for these activities are the participants of your Design Thinking process. Useful techniques include, for example, mind mapping (see Section 3.3.5) and storyboards (see Section 3.2.4), which help to restructure and visualize the results.

Ideate:

Explore a wide variety of possible solutions through generating a large quantity of diverse solutions, allowing you to step beyond the obvious and explore a range of ideas.

From a Requirements Engineering perspective, this phase is a creativity phase in which you define elicitation activities with elicitation objectives that focus on the generation of a large number of different ideas.

The requirements sources for these activities are the participants in your Design Thinking team. Design Thinking emphasizes the importance of generating several different ideas to solve the problem at hand. The result quality is flexible and is driven by the available resources and time limits. Useful techniques include, for example, brainstorming (see Section 3.2.1).

Prototype:

Transform your most promising ideas into physical form so that you can experience and interact with them and, in the process, learn and develop more empathy.

From a Requirements Engineering perspective, this phase is about learning from the development of prototypes. You define elicitation activities with elicitation objectives that focus on the elaboration of ideas from the ideate phase. The requirements sources for these activities are the participants in your Design Thinking team. The result quality depends on the project and especially on the available time and resources. If your project has a low budget and hard time constraints, you should develop cheap and simple prototypes. Design Thinking provides a useful rule of thumb for prototypes: Do not spent too much effort on a prototype, otherwise you can cling too much to a certain idea and introduce biases in the evaluation. Useful techniques are all types of prototyping (see 3.2.3), for example paper-and-pencil prototypes, mock-ups or wireframes.

Test:

Try out high-resolution prototypes (see Section 3.2.3) and use observations and feedback to refine prototypes, learn more about the user and refine your original point of view.

From a Requirements Engineering perspective, this phase is a mixture of requirements validation and elicitation. You define elicitation activities with elicitation objectives that focus on getting feedback from stakeholders. The requirements sources for these activities are stakeholders outside of your Design Thinking team. This neutral feedback is important, since the stakeholders in your team were involved in the development of ideas and could be biased. The result quality should be high, since the feedback from the stakeholder is very important to further improve the developed ideas. Useful techniques include, for example, quantitative interviews (see 3.1.1.1), and workshops (see 3.1.1.3) in which the prototypes are presented.

1.4.4.4 Instantiation

To start a Design Thinking process, you first have to identify your Design Thinking team and to plan the empathize phase in detail. At the beginning, your plan could look like this:

No.	Elicitation Objective	Elicitation Technique	Requirements Source	Result Quality
Short Term Activities				
1	Identify Design Thinking Group members	Interview	project sponsor / head of technical department / ...	High certainty / high completeness
2	Understand people in system context: User (empathize phase)	Interview	User of system	High certainty
3	Understand people in system context: user	Observation	User of system	High certainty
4	Understand people in system context: superiors of users	Workshop	Superior of users	High certainty
Long Term Activities				
5	Define Phase	Workshops	Design Thinking Team	TBD
6	Ideate Phase	Creativity Techniques	Design Thinking Team	TBD
7	Prototype Phase	Prototyping	Design Thinking Team	TBD
8	Test Phase	TBD*	TBD*	TBD*

*The TBDs in this table mean that you cannot or should not decide these aspects in the beginning of the Design Thinking process because they are highly dependent on the outcomes of the other phases. Keep in mind also the iterative nature of the Design Thinking process and that the long-term activities are defined in an abstract way and have to be detailed in the course of the process.

1.4.4.5 Further reading

[Brown2009] gives broad overview of the ideas behind Design Thinking and its potential.

[Design Council 2007] explains the Double Diamond model that was published in 2005 by the British Design Council.

[KnZK2016] present a compact version of Design Thinking in five days. The book is structured along the five days and gives concrete methodological guidance on what to do on each day.

[LeLL2018] provides a useful collection of hints, examples and techniques for Design Thinking professionals.

2. Requirements Sources

This chapter covers the three types of requirements sources and how they can be identified, classified and managed. Section 2.1 explains the pragmatic and systematic approach for the identification of requirements sources. Section 2.2 covers the identification, classification and management of stakeholders and Sections 2.3 and 2.4 do the same for documents and systems.

2.1 Fundamentals of requirements sources

The quality and completeness of requirements depend greatly on the requirements sources involved. Missing a relevant source will lead to an incomplete understanding of the requirements and increases the risk of your endeavor. For example, a lack of user involvement increases the risk of low acceptance of the new system, neglecting an important stakeholder can lead to the project being blocked in a critical phase, and ignoring the existing functionality of a legacy system can lead to overlooking basic requirements. Hence, in the course of development, the Requirements Engineer has to identify and consult all relevant requirements sources.

The three most important types are [IREB2017]:

- Stakeholders
- Documents
- Systems

As no requirement comes without a source, it is naturally one of the first activities in requirements elicitation to identify the potential requirements sources. This is an iterative process: it is not enough to only identify these sources at the beginning of a project or product development, but rather a process that must be repeated over and over again. With each new elicitation activity and an increasing knowledge of the product to be built, new potential requirements sources may be identified, leading in turn to further elicitation activities and source identification in a recursive process.

Example

You are interviewing a potential user (time traveler) of the time machine you are about to develop. During the interview, the time traveler mentions that her colleague usually performs the action you've just asked about and that she would have to look up the details in the time travel process documentation.

The time traveler has just revealed two potential requirements sources you should take into consideration.

So, you ask for the colleague's name and for details on the time travel process documentation she just mentioned.

After the interview, you check in your requirements sources documentation (see 2.3.2) whether the interviewee's colleague and the time travel process documentation have already been identified, or if they are new potential requirements sources.

By interacting with a previously identified requirements source you have identified another one. This makes it a recursive process.

Be prepared to identify new requirements sources even during late project stages, when, usually, a sense of "we know who they are and what they want" is lingering over the project participants.

You should constantly revisit your requirements sources documentation and re-evaluate whether it is still up to date, or if new requirements sources may be relevant or some previously identified sources have turned out to be obsolete.

Pragmatic and Systematic Identification of Requirements Sources

We distinguish two basic approaches for the identification of requirements sources:

- Pragmatic identification
- Systematic identification

Pragmatic Identification of Requirements Sources

"Pragmatic Identification" is a fancy word for using your intuition and experience. If you have been involved in a project within the same department before, or in a similar project in a different business context, you will be able to name a list of potential stakeholders, documents and systems without much thinking.

This is an important means of identification of potential requirements sources; with a small investment in time you gain a list of potential requirements sources to start with. Later in product development you may come up with further potential requirements sources.

It is dangerous, however, to rely on pragmatic identification as the only approach, as you might miss crucial requirements sources. You should always back up and add to your results from pragmatic identification by use of systematic identification.

Systematic Identification of Requirements Sources

Systematic identification of requirements consists of two steps:

1. Determine criteria that can characterize the requirements sources and thus contribute to their identification.
2. Systematic search based on the criteria (apply the snowball principle[3])

For systematic identification, you need to define elicitation activities (see 0) with elicitation objectives focused on the identification of requirements sources. This way, you break down the abstract objective of finding all relevant requirements sources into specific, actionable tasks.

Figure 8 through Figure 10 show examples for elicitation activities for the identification stakeholders for building a time machine.

ID	**RS_EA_01**
Elicitation objective	Find at least 10 potential time travelers (users)
Result quality	User name, role and contact data
Requirements source(s)	Organization chart
Elicitation technique	Perspective-based reading

Figure 8: Example for an elicitation activity for the identification of users as stakeholders

[3] When you identified a new stakeholder, ask him or her for further requirements sources.

ID	RS_EA_02
Elicitation objective	Find at least 5 legal documents potentially relevant for a time machine
Result quality	Name of document, current version, place where to find it
Requirements source(s)	Evelyn Hall, Bob Miller, and the developing company's legal advisor/lawyer
Elicitation technique	Interview

Figure 9: Example for an elicitation activity for the identification of legal documents

ID	RS_EA_03
Elicitation objective	Find out: who is the developing company's legal advisor/lawyer
Result quality	Name, contact data, availability
Requirements source(s)	Legal department
Elicitation technique	Telephone interview

Figure 10: Example for an elicitation activity for the identification of a legal advisor or lawyer

In this example, while planning the requirements sources (RS) elicitation activity (EA) RS_EA_02, the Requirements Engineer realized that she had not yet identified who the company's legal advisor is for the development project. Therefore, she added another elicitation activity (RS_EA_03) to address that issue.

You should always apply both pragmatic as well as systematic identification. Pragmatic identification saves time and resources but bears the risk of subjective bias. In most cases, pragmatic identification will not be sufficient on its own. Systematic identification, on the other hand, requires more time and resources than pragmatic identification, but also leads to a more objective, and potentially more complete, list of requirements sources.

We suggest starting with pragmatic identification and then to back up and amend your results using systematic identification.

2.2 Identify, classify, manage stakeholders

This Section focuses on the main requirements source for most projects: the stakeholders. Stakeholders are human beings or legal persons in the context of your project who influence the project and who are affected by it. As a Requirements Engineer, you can meet them, observe them, talk to them, and ask them questions. Legal persons are represented by natural people (e.g. the WeAreTheBest Ltd company is represented by its CEO or a spokesperson).

Of course, stakeholders can be aggregated into groups or roles, such as users, maintenance staff, testers, etc. This concept of stakeholder groups or roles is very useful for many Requirements Engineering tasks. Remember, however, that you cannot interact with abstract roles, but ultimately always need a real human being to communicate with.

And you don't usually get in trouble with abstract groups, but with individuals (e.g. Mr. Fudge, the head of Travel Authorization, who was not invited to give input on the requirements for the Time Travel system).

Hint 2.2.1:

It is good practice to name individual stakeholders for each stakeholder group, especially for important internal groups, such as the legal department, security officers, managers, representatives from related departments, etc.

Section 2.2.1 explains how stakeholders can be identified systematically and pragmatically;
Section 2.2.2 deals with stakeholder relationship management (including classification);
Section 2.2.3 covers stakeholder documentation and
Section 2.2.4 focuses on the user as a specific stakeholder role.

2.2.1 Identifying and selecting stakeholders as requirements sources

Stakeholders play a vital role in most development projects. Identifying and managing them is therefore a major task during elicitation of requirements.

2.2.1.1 Pragmatic approach to identification of stakeholders

It is common to identify stakeholders pragmatically. From his/her experience in the project context (e.g. from previous projects in the same organization) and by reuse of existing stakeholder lists, the Requirements Engineer quickly assembles an initial list of stakeholder groups and roles (this often already includes individual representatives).

2.2.1.2 Systematic approach to identification of stakeholder groups and roles

Although pragmatic identification is a useful way to start stakeholder identification, it should always be backed up with systematic stakeholder identification.

Helpful sources and techniques for systematic identification of stakeholders are:

- **Checklists of typical stakeholder groups and roles**
  Lists like the metaphoric stakeholder country map from [Rupp et al.2014] or the following simple list can be used for any project as a guideline:
  - Direct system users,
  - Business / process managers,
  - Clients and individual customers or customer-representing organizations,
  - Opponents and competitors,
  - IT staff,
  - Governmental and regulatory institutions.

  Look at every stakeholder role in the list and clarify whether or not it is a relevant role for your project. Additionally, it makes sense to create your own checklist over time, in particular if you are working in a specific field (e.g. insurance industry), where the stakeholder roles required for each project may be very similar.
- **Organizational structures**
  (e.g. organization charts of the company that will use the system to be built)
  Most companies have organization charts. They can be useful to find the different departments in a company and the stakeholders in those departments.

- **Business process documentation**
  Although they may not be entirely up to date, many companies have documented their business processes in some way. That could be a business process model (e.g. BPMN model) or a natural language description of how the process works. From such documentation, you might be able to find out which role is responsible for a specific task. Such documentation is itself a potential requirements source (see 2.3).
- **Stakeholder categorization schemata**
  Existing stakeholder categorization schemata are also valuable sources for the identification of potentially relevant stakeholder roles. They name typical categories and roles of stakeholders.
  Examples are: Alexander's onion model [Alexander2005] – see subsequent section for more details – or Robertson's generic stakeholder map [RoRo2013].
- **Information focus**
  Try to identify relevant information or fields of knowledge and derive stakeholder groups who can provide that information or who are experts in those fields.

  Example:

ID	RS_EA_04
Elicitation objective	Find at least two stakeholders who can potentially provide me with information about the physical constraints of time travelling.
Result quality	Stakeholder name, role and contact data
Requirements source(s)	Organization chart
Elicitation technique	Perspective-based reading

Figure 11: Example for an information-focused elicitation activity

- **Product lifecycle analysis**

  Take a virtual stroll through the lifecycle of the product to be developed. Who will interact with the product or its documentation in any way from its requirements through its development and usage to its deinstallation or deconstruction?
  This will reveal other departments (e.g. production and maintenance) or organizations (e.g. secondhand users, suppliers or recycling companies) as potential stakeholders.

Hint 2.2.2:

It has proven useful to hold a workshop for stakeholder identification, gathering all potential stakeholder roles, names and fields of knowledge that come to the participants' minds. As a next step, the triplets of name, stakeholder role(s) and field(s) of knowledge, as well as potential gaps, are identified.

2.2.1.3 Systematic approach to identification of individual stakeholders (persons)

As noted before, it is important to identify contactable, tangible individuals. Such stakeholders have names, contact data and birthdays. The latter may not be of prime interest for Requirements Engineering activities, but it is worthwhile information when it comes to stakeholder relationship management.

During systematic individual stakeholder identification, the Requirements Engineer defines corresponding elicitation activities.

Example:

ID	RS_EA_05
Elicitation objective	Find at least one person for the stakeholder role “Time traveler training”
Result quality	Stakeholder name and contact data
Requirements source(s)	Dr. Emmet Brown (experienced time traveler, he might know suitable candidates)
Elicitation technique	Interview

Figure 12: Example for an elicitation activity to identify individual stakeholders

2.2.2 Stakeholder relationship management

Problems with stakeholders typically arise if the rights and obligations of a stakeholder, in respect to the proposed system or the current project, are not clear or if the stakeholder’s needs are not sufficiently addressed. Stakeholder relationship management is an effective way to counter problems with stakeholders. In order to engage stakeholders in the elicitation process, we need to ensure that they know what the project is about and what their role within the project is.

The stakeholder circle [Bourne2015] is a helpful framework for successful stakeholder relationship management. It consists of the following five steps:

1. Identification of all stakeholders
2. Prioritization to determine who is important
3. Visualization to understand the overall stakeholder community
4. Engagement through effective communication
5. Monitoring the effect of the communication

Bourne’s framework is aimed at project stakeholder management. We have slightly modified the details of the individual steps to fit the context of requirements stakeholder management and the IREB syllabi.

Step 1: Identification of all stakeholders

The stakeholder circle starts with the identification of all stakeholders. This step is described in Section 2.2.1.

Step 2: Prioritization to determine who is important

Not all stakeholders are equal. Some are more important than others and some are more important at the beginning of a project than at the end.

As we have limited resources available for stakeholder management and requirements elicitation, it is vital to prioritize the identified stakeholders. Depending on the project, different prioritization schemata may make sense. Before choosing a prioritization schema, ask yourself:

- What is the benefit of this schema?
- How is the information it provides helpful for the project?
- What is the advantage of this schema over the other schemata we could use?
- How does it help to identify who is really important for our project?

The first step before prioritization is classification. When all stakeholders are classified according to a certain schema, then those classes can be prioritized with respect to each other. Often it also makes sense to further prioritize within one identified class.

One example for a classification schema is Ian Alexander's Onion Model [Alexander2005]. It uses three classes:

- **Stakeholders of the system:**
  These stakeholders are directly affected by the new or modified system. Typical examples of that class are users, maintenance staff and system administrators.
- **Stakeholders of the containing system:**
  These stakeholders are indirectly affected by the new or modified system. Typical examples of that class are managers of users, project owners or sponsors.
- **Stakeholders of the wider environment:**
  These stakeholders have an indirect relationship to the new or modified system. Typical examples of that class are legislators, standard setting bodies, non-governmental organizations (NGO, e.g. unions or environment protection associations), competitors and also the project members, who are involved in the development of the system but will not be affected by the system in productive use.

Stakeholders can also be classified according to their influence on the project (high versus low influence) and their motivation in relation to the project (strong versus weak motivation). Stakeholders with large influence can, for example, either obstruct the project, or advance it. Stakeholders with higher motivation are, for example, valuable to the project, because they themselves have an interest in advancing the project [Rupp et al.2014].

Further examples for classification attributes of stakeholders are:

- Proximity
- Availability
- Interest
- Power
- Experience in similar projects
- Communication skills

Step 3: Visualization to understand the overall stakeholder community

All information gathered about the stakeholders must be documented (i.e. visualized) and that documentation has to be kept up to date. Such a visual representation of stakeholders helps in understanding the overall stakeholder community and to ensure that no important stakeholder is overlooked.
See Section 2.2.3 for details on visualization/documentation of stakeholders.

Step 4: Engagement through effective communication

Now it is important to understand what each stakeholder expects in terms of communication and what their attitude towards the project is. The following elements should be considered [Bourne2015]:

- Culture (organizational, team or individual)
- Identification with the (development) activity and its outcomes,
- Perceived importance of the activity and its outcomes
- Personal attributes, such as personality and role.

After assessing the stakeholders' attitudes, you need to define a "realistic target attitude" for each of them. That target attitude should serve both the stakeholder and the development project.

Is there a gap between the current and target attitude? If yes, you should define activities aimed at closing that gap (communication plan).

The following questions may be helpful for the engagement step:

- How often should I contact specific stakeholders to inform them about the status with respect to the elicitation of requirements?
- Which information is relevant for which stakeholder?
- What is the best way to keep a specific stakeholder up to date? (e.g. phone call, email, newsletter, lunch.)
- What is the best way to contact this stakeholder if I need information?
- How will I keep track of when I last contacted this stakeholder?
- Are there any cultural factors relevant for communication?

Hint 2.2.3:

Requirements stakeholder management and project stakeholder management tend to overlap. Make sure to coordinate with the person responsible for project stakeholder management on how you are going to deal with this overlap.

Requirements stakeholder relationship management is focused on managing the relationship to get the necessary requirements from a stakeholder. Project stakeholder relationship management usually has a broader view on stakeholder relationship management.

Step 5: Monitoring the effect of the communication

Repeat the assessment of step 4 regularly to identify where further stakeholder relationship management activities are required and to see whether the activities taken have been effective. Adjust the communication plan as required to keep closing the gap between current and target stakeholder attitudes.

2.2.3 Documentation schema for the stakeholders involved

All information gathered during stakeholder identification and stakeholder relationship management must be documented sufficiently. Such documentation should include at least the following for each stakeholder:

- Name
- Function (role)
- Contact data
- Availability (spatial and temporal)
- Relevance
- Area and extent of expertise
- Goals and interests regarding the project

Furthermore, the classification information (e.g. according to Ian Alexander) and priority from step 2, as well as information gained from step 4 (including the communication plan), should be documented.

Depending on the project, additional information may be relevant. Influencing factors could be:

- *Public relevance*: In a context with higher public relevance, it may be useful to document how much a stakeholder knows or can influence public opinion.
- *Time criticality*: In a context with a very strict time frame, the availability or response time of a stakeholder can be very important information when critical decisions are to be taken.

Make sure to use a form of documentation that fulfils the documentation and stakeholder relationship management needs of your project setting.

Commonly used forms of documentation are:

- Stakeholder table
- Stakeholder database (often incorporated in the requirements management tool)
- Stakeholder mind map

Additionally, diagrams or other graphical representations can be used to track changes in stakeholders‘ attitudes or priorities.

The stakeholder documentation has to be kept up to date as long as information on stakeholders might be needed (i.e. usually at least until the end of the development project, or even until the end of the product lifecycle).

2.2.4 The user as a special stakeholder group

For interactive systems with a user interface, all direct users of the system are of prime interest for the Requirements Engineer.

In-house users (in-company, individually known and involved) are significantly different from outside users (e.g. buyers of consumer products; outside of the company, generally not known individually and not directly involved).

Usually, the number of potential users does not allow involving all individuals in the elicitation process. For this reason, the actual users can be aggregated into user groups, based on user analysis or on the domain knowledge of other stakeholders.

A common way to represent user groups is the use of personas [Cooper2004]. Personas are fictitious individuals, representing typical user groups of the system with similar needs, goals, behaviors or attitudes. Personas are modeled from data collected about real users through user research [BaCC2015]. If no relevant user research data is (yet) available, provisional personas, also called ad-hoc personas [CRCN2014] or proto-personas [Gothelf2013], can be created.

Personas can also be created based on raw data gathered by contextual inquiry, interviews, surveys or apprenticing (refer to the description of these methods in Section 2.2.2), as described by [Goodwin2009]. The central concept of building personas is the identification of bipolar variables that characterize and differentiate personas.

The resulting user groups or personas should be prioritized as primary or secondary. The system, especially its user interface, will be optimized for the primary user group. Secondary user groups are only supported in so far as serving their needs does not compromise the user experience of the primary ones.

If users are your primary stakeholders, then apply the human-centered design pattern to meet their expectations (see Section 1.4.3).

Hint 2.2.4:

The agile approach to software development focuses on the value the solution shall provide to users ("user story"). Therefore, the user of an interactive system is regarded as a primary stakeholder. Nevertheless, there is a risk, particularly in large organizations, that agile teams or product owners are not in contact with the real users. Personas may have been invented by somebody pretending to know the users! In such a context, stakeholders of the organization (managers, business departments, legal department, marketing, etc.) are much more present and dominant than the external end user. Be aware of this trap: Insist on direct access to end users to perform proper user research and on collecting direct feedback after sprint-delivery in order to really get to know your users and their needs.

2.3 Identify, classify, manage documents

Documents are used to transfer information between humans over time and distance. Often, requirements can be derived from documents, making them a valuable source.

In Section 2.3.1 we discuss the identification and selection of such documents, and in Section 2.3.2 we cover the documentation of documents as requirements sources.

2.3.1 Identifying and selecting documents as requirements sources

Requirements can be derived from many different types of documents. Depending on the particular character of the development project, documents may have either high, medium or low significance as requirements sources. Typically, technical systems engineering projects (e.g. dual-clutch transmission) have many documents as requirements sources (e.g. technical standards, patents), whereas human-centric projects (e.g. a mobile shopping app) have less.

Possible types of documents that may be used as requirements sources are (non-exhaustive):

- **Technical standards, legislation, internal regulations**
  All development projects have legal constraints under both civil and criminal law (national and potentially also international). Typically, data privacy laws are of relevance for many development projects. Within the medical and food industries, U. S. Food and Drug Administration (FDA) regulations are of high importance, while in the automotive industry specific international standards, like ISO 26262 on functional safety, apply. These are just a few examples; each industry has its own set of applicable standards and laws. In addition, each company potentially has internal regulations that may contain requirements for the development project (e.g. style guide).
- **Requirements documents**
  Not all systems are developed as green field projects. Often, one or more predecessor systems are to be replaced or a variant of an existing system is developed (e.g. a new contract management system for an insurance company that has recently merged with a competitor).
  In such cases, requirements documents from predecessor systems may be valuable sources for new requirements, e.g. business rules, glossary, business object models, use cases, workflow specifications, authentication and authorization documents, etc. Requirements documents from interfacing systems may also be relevant.

- **User manuals**
  If predecessor or competitor systems exist, user manuals may be available. They give a good overview of system functionality and may be more up to date[4] than the requirements documentation of those systems.
- **Strategy papers**
  If the system to be developed is part of an overall company strategy, strategy papers or presentations may exist that could include relevant requirements for the system.
- **Goal documentation**
  Before a system development is started, several documents usually already exist that evaluate the necessity and potential benefits of the project. Such documentation usually includes the goals of the project, which are an essential requirements source.
- **Business process documentation**
  If the system to be developed relates to one or more steps of a business process, any business process documentation is a valuable source for requirements. Such documentation can be found in different forms: e.g. plain text documents, slide presentations, intranet pages, wiki sites or BPMN or other models.
- **Interface documentation**
  Most systems interact with other systems. Any interface documentation for those interfacing systems may contain relevant information for the system to be developed.
- **Documents generated by the business process**
  In many business processes documents such as insurance contracts, bills or lists of clients are generated. Such documents often contain valuable information for any system developed to support that process.
- **Documents generated in technical analysis**
  Systems development including hardware or mechanical components often involves analyzing the results from simulations, safety assessments or established methods like Quality function deployment (QFD), Failure Mode and Effects Analysis (FMEA).

As with stakeholders, documents can be identified pragmatically and systematically.

When *pragmatically identifying* documents, Requirements Engineers use their current knowledge and experience of the context (e.g. domain) to name relevant documents and document types.

In *systematic document identification*, the Requirements Engineer can:

- Search for representatives of typical document categories
  (e.g.: Which technical standards apply for a time machine?)
- Search for references in previously identified documents to other relevant documents
  (e.g.: Does ISO26262 cite other standards that may apply for our time machine?)
- Ask previously identified stakeholders for relevant documentation
  (e.g. Ask Dr. Emmet Brown whether there is any documentation on the Flux Compensator)

[4] Good requirements management should avoid that scenario. Unfortunately, not all companies and projects have reached such a level of requirements management maturity, yet.

- Search for documentation on systems previously identified as relevant (see 2.4)
  (e.g. Is there a user manual for the TARDIS or for the DeLorean Time Machine?)

During *systematic document identification*, the Requirements Engineer defines elicitation activities focused on the identification of documents. Two different types of elicitation objectives have to be considered:

- *Information-focused:* Finding documents for certain information required
- *Document-focused:* Finding documents of certain document types, considered relevant for the development

ID	**RS_EA_06**
Elicitation objective	Find at least one document on the physics of time travelling.
Result quality	Document name, author, date published, place to find it
Requirements source(s)	Library, the Internet
Elicitation technique	Perspective-based reading

Figure 13: Example for an information-focused elicitation activity

ID	**RS_EA_07**
Elicitation objective	Find out which legal documents affect time travelling
Result quality	Document name, publishing authority, date published, region in which applicable, where to find it
Requirements source(s)	Dr. Who, the developing company's legal advisor/lawyer
Elicitation technique	Interview

Figure 14: Example for a document-focused elicitation activity

The Requirements Engineer needs to decide which of the collected documents have what potential value as requirements sources. Thus, the documents have to be scanned and evaluated as to their potential usefulness.

Depending on the context, different criteria may be relevant. Apart from content, other aspects may influence the value (and hence priority) of a document as a requirements source:

- **Availability**: Documents may be confidential and require certain security clearance to be accessed.
- **Size**, or even better, estimated size/content ratio: If a project has a tight deadline, small documents containing important information may be more valuable than long documents with many irrelevant details.
- **Age**: The older a document, the higher the probability that its content is out of date.
- **Relevance**: The requirements document of the predecessor system may be more relevant than the requirements document of an interfacing system.

2.3.2 Documentation schema for documents

All the information gathered on documents must itself be sufficiently documented. Such documentation should include at least:

- Title
- Place where it is kept (e.g. physical folder name, link to digital document)
- Version of the document
- Short description (what kind of information the document can provide)
- Relevance

Depending on the context, additional information may be relevant. Examples are:

- Person responsible for the document (document owner)
- Person who has added the document to the list (relevant if more than one person is updating the list)
- Date when the document was added to the list
- Date when the document was last reviewed (for requirements or whether a new version is available)
- Size of document

Documents always have certain relationships to stakeholders, which should also be recorded. Thus, it may make sense to link the documents documentation with the stakeholder documentation.

The following are examples of relationships amongst documents and stakeholders:

- Stakeholders mentioning the relevance of the document
- Author, issuing organization
- Organizations using the document in their processes
- Organizations involved in verifying the adherence to the document

The Requirements Engineer has to keep the information about documents up to date. This includes reconsidering whether additional documents have become relevant or documents identified earlier have lost relevance. Special attention should be given to changes, updates and version numbering.

2.4 Identify, classify, manage systems

Within the context of the system to be developed, other systems may exist that represent requirements sources. As soon as a system is identified as a potential requirements source, it becomes part of the system context (see [IREB2017]).

In Section 2.4.1 we discuss the identification and selection of such systems, and in Section 2.4.2 we cover the documentation of systems as requirements sources.

2.4.1 Identifying and selecting systems as requirements sources

Requirements can be derived from many different types of systems. Depending on the character of the development project, other systems may have a high, medium or low significance as requirements sources. For systems with many interfaces there are likely to be many systems that are relevant as requirements sources. If a new system is developed to replace one or more existing systems, those systems will have a high significance as requirements sources.

Even for new systems, existing systems from other fields of application may be relevant (also see Section 3.2.2 analogy technique).

Types of systems that may be relevant as requirements sources are (non-exhaustive):

- **Interfacing systems including legacy systems**
  For the new system to be able to interact with its interfacing systems, it is essential to know those systems, and in particular their interface specifications/requirements.
- **Systems sharing a platform / environment / ecosystem**
  If the system to be developed is to be integrated into an existing platform, environment or ecosystem, then there will be requirements (mostly constraints) resulting from that platform, environment or ecosystem, for example: a usability concept, or technical requirements.
- **Competitor systems**
  Competitors are constant drivers for innovation. It is therefore very important to know competitors‘ products, both OTF (off the shelf) products as well, as those used or being developed within a company for example by other departments or offices. In the case of the latter, we should look out both for potential synergies as well as for solutions might become internal competitors.
- **Systems with similar data, functionality or user interfaces**
  Hardly any system is unique. In fact, it might not be desirable to develop a unique system.
  Systems that are similar to your system under development, whether because they process similar data (e.g. vending machine vs. cash register vs. online shop), have similar functionality (e.g. train ticket vending machine vs. beverage vending machine vs. ATM) or share a similar user interface (e.g. remote control vs. calculator) may be valuable sources for requirements.
- **Predecessor system(s) to be replaced**
  If a predecessor system is to be replaced, the old system is usually one of the most important requirements sources. It is important, though, not to become too biased by the old system, and to be open for improvements and amendments.
- **Future systems (under construction)**
  It is important to watch out for future systems in the context of the system to be developed. There could be systems already in development. Such future systems may change the environment of the system to be developed and hence its requirements.

As with stakeholders and documents, systems can be identified pragmatically and systematically.

When *pragmatically identifying* systems, Requirements Engineers use their current knowledge and experience of the project and its context (e.g. domain) to name relevant systems and system types.

In *systematic identification*, the Requirements Engineer can:

- Use the system context documentation
- Ask previously identified stakeholders about relevant systems
  (e.g. Which systems are planned or are being developed that might be relevant for the system to be developed?)
- Search previously identified documents for information on relevant systems
  (e.g. search the systems architecture documentation of an interfacing system for information about its interfaces to other systems)

- Use idea-generating techniques to identify potentially analogous systems
  (e.g. brainstorming)
- Conduct market research to identify competitor systems
  (i.e. which other systems are available or being developed that serve the same purpose?)
- Consider legacy systems
  (e.g. Which legacy systems served a similar purpose?)

During *systematic system identification*, the Requirements Engineer defines elicitation activities focused on the identification of systems. Two different types of elicitation objectives have to be considered:

- *Information-focused*: Finding systems that contain certain required information
- *System-focused*: Finding systems of certain types that are considered relevant for the development project

Figure 15 and Figure 16 give examples for information-focused and system-focused elicitation activities.

ID	**RS_EA_08**
Elicitation objective	Find at least five competitor time travel machines.
Result quality	System name, creator, place where (or when) to find it. Ideally, the time machines should be working, otherwise non-working time machine would be ok, too.
Requirements source(s)	Library, the Internet, time travelling conference
Elicitation technique	Perspective-based reading

Figure 15: Example for a system-focused elicitation activity

ID	**RS_EA_09**
Elicitation objective	Find at least five systems that provide space for two adults.
Result quality	System name, creator, place where to find it
Requirements source(s)	All developers
Elicitation technique	Workshop

Figure 16: Example for an information-focused elicitation activity

2.4.2 Documentation schema for systems

All information gathered on systems must be documented sufficiently. This should include at least each system's:

- Name
- Type (e.g., competitor system, predecessor system, interfacing system, ...)
- Data, functionality, processes, user groups (brief description)

- Version
- Relevance

Depending on the context, additional information may be relevant. Examples are:

- Person responsible for the system (system owner)
- Person who has added the system to the list (relevant if more than one person is updating the list)
- Date when the system was added to the list
- Date when the system was last reviewed (for requirements or whether a new version is available)
- Number of users

Special attention should be paid to directly interfacing systems. They can be categorized as:

- Data sources: providing data
- Data sinks: using data
- Supporting systems, like an operating system (OS) or Database Management System (DBMS)

Systems always have certain relationships with the stakeholders, which also should be recorded. Thus, it may make sense to link the systems documentation with the stakeholder documentation.

The following are examples of relationships among systems and stakeholders:

- Stakeholders/Organizations that use the system in direct or indirect ways in their processes
- Stakeholders/Organizations that operate the system
- Stakeholders/Organizations that design, develop, or market the system
- Stakeholders/Organizations that maintain the system, offer support or training
- Organizations that observe the system (e.g. governments, NGOs)
- Stakeholders, mentioning the relevance of the system
- System owner
- Organizations involved in verifying the adherence of the system to applying laws and standards

The Requirements Engineer has to keep the information about systems up to date. This includes reconsidering whether additional systems have become relevant, or whether systems identified earlier may have lost relevance. Keep an eye on changes, updates and version numbering.

Information about systems is usually present in documents. These documents should be managed separately as requirements sources (see 2.3). The relation between document and system should also be documented.

3. Elicitation

A plethora of techniques has been developed for the elicitation of requirements. For the Advanced Level Elicitation syllabus, and therefore for this handbook, we have selected some of those techniques and present them here in a structured way.

As we have added techniques to the Advanced Level, we have at the same time adapted the categorization used in the Foundation Level. The categorization makes the presentation of our 19 elicitation techniques and thinking tools more comprehensible. This differentiation is, of course, an artificial one: in practice, there is no clear separation between the techniques. However, for presentation and teaching purposes, the differentiation is important in providing a structure and in describing the primary focus of each technique.

All elicitation techniques described in this chapter are structured as follows:

- **What is it?**
  describes briefly the main factors of the elicitation technique.
- **Role of participants (if applicable)**
  describes which roles are involved when applying this technique.
- **Preparation**
  describes which actions have to be taken to prepare the application of the elicitation technique.
- **Application**
  describes how the elicitation technique is applied and what has to be taken into consideration during its application.
- **Result processing**
  describes which actions are required to process the results (see note below).
- **Typical artifacts**
  names the artifacts typically used or produced during application of the elicitation technique (see note below).
- **Opportunities**
  describes the possible advantages of the elicitation technique.
- **Challenges**
  describes the possible pitfalls or disadvantages of the elicitation technique.
- **Variants (if applicable)**
  describes variants of the elicitation technique and provides a brief description.

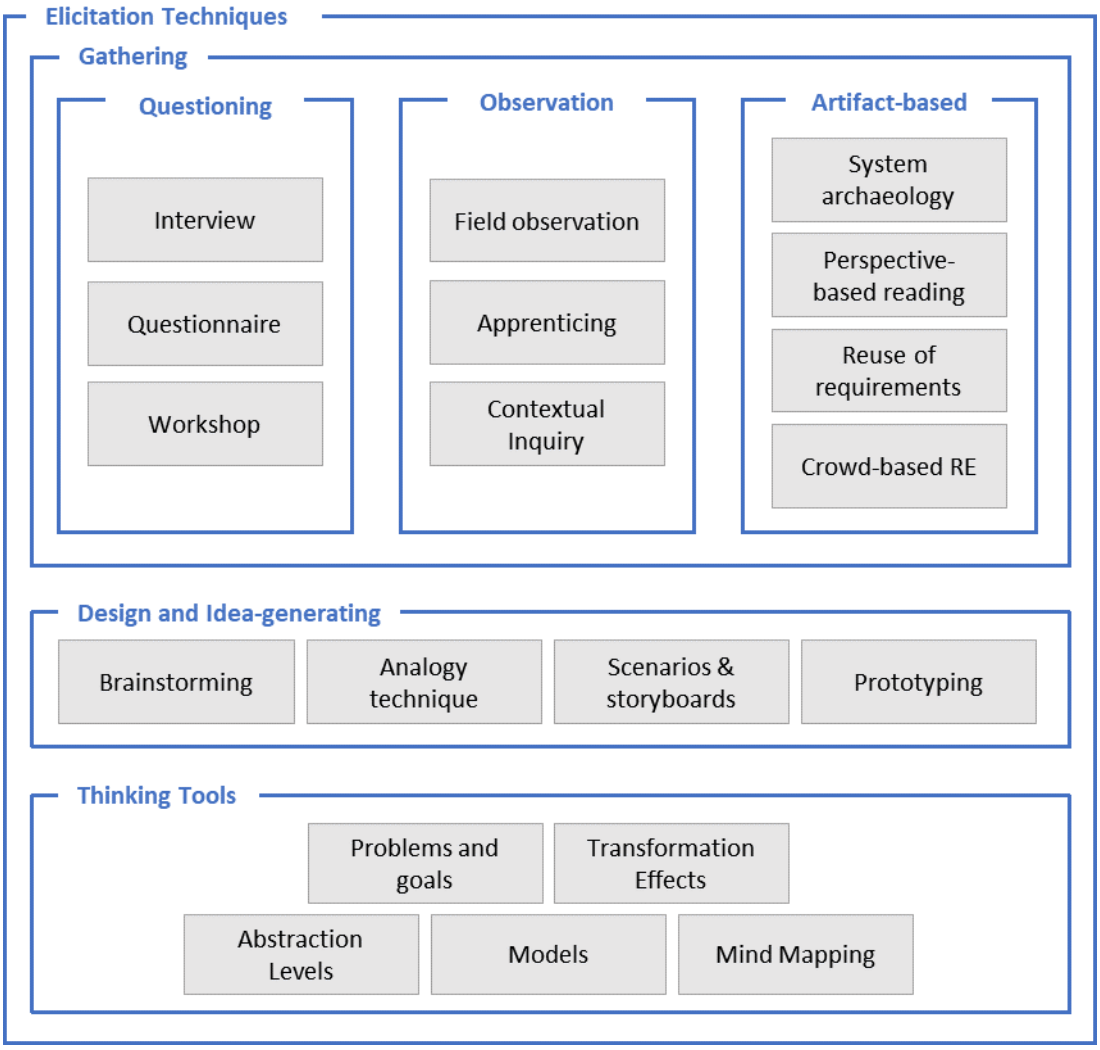

Figure 17: Overview of elicitation techniques

Note:

The results acquired on performing an elicitation technique are raw information that has to be processed and formulated into requirements. The latter is part of requirements documentation and as such not covered in this Advanced Level Elicitation Handbook.

After each elicitation activity, the documentation has to be updated with the information received, e.g. requirements documentation, system context documentation, requirements resources documentation. It should also then be decided whether further elicitation activities are required.

Section 3.4 provides typical identifying characteristics of elicitation techniques. These may be used to describe new techniques and to give general guidelines as to which characteristics are useful in a given project situation.

3.1 Gathering techniques

Gathering techniques are established techniques for requirements elicitation. They help to elicit satisfiers and dissatisfiers. We use this term to group questioning techniques (interview, questionnaire, requirements workshop), observation techniques (field observation, apprenticing, contextual inquiry) and artifact-based techniques (perspective-based reading, system archaeology, reuse of requirements, crowd-based Requirements Engineering).

3.1.1 Questioning techniques

Questioning techniques aim to pose appropriate questions to stakeholders in order to receive answers from which requirements can be derived.

Open-ended and Closed-ended Questions

For all questioning techniques covered in this Section, it is important to know the distinction between open-ended and closed-ended questions.

For closed-ended questions, the answers are given or defined, either by the question itself or by explicit reference to the available answers (e.g. How old are you? Do you like fish?). Closed-ended questions lead to quantitative data, i.e. data that can be processed statistically without further adaptation.

Open-ended questions, on the other hand, allow a free response, in order to query narrative or argumentative knowledge, and therefore lead to qualitative data (e.g. What is your favorite dish? How do airplanes fly?). Qualitative data can only be evaluated by cognitive analysis.

In the context of requirements elicitation, predominantly qualitative data is required. Therefore, mainly open-ended questions are needed. However, there are also situations requiring quantitative data (e.g. evaluating whether the statement of one representative user is confirmed by other users). Here, closed-ended questions should be used.

When using questioning techniques in an elicitation activity, the quality result definition should include whether qualitative or quantitative data should be elicited.

In the following, the three most relevant questioning techniques are described: interview, questionnaire and requirements workshop.

3.1.1.1 Interview

Due to their flexibility, interviews are probably one of the most frequently-used elicitation techniques. They do not require a big setup, or specific tools, and can be used to elicit high-level requirements as well as very specific ones. Typically, requirements elicited with an interview are satisfiers, as the interviewee voices conscious information. By using the right questioning techniques and observing the user's reactions, however, dissatisfiers or delighters may also be identified.

Although an interview is not very complicated and most people have a good understanding of what it is, it requires clear goals and detailed preparation in order to make good use of the interview time and obtain valuable and sustainable results.

What is an interview?

In an interview, the Requirements Engineer asks one or more stakeholders questions in order to elicit new requirements or to refine existing ones (elicitation objective)**.** The stakeholders answer these questions and the Requirements Engineer (or an assistant) records the information he/she receives. An interview differs from a plain conversation due in main to three factors: the role of the participants, preparation and result processing.

Role of participants

In an interview, three roles are mandatory: *interviewer*, *interviewee* and *note-taker*.

The Requirements Engineer takes the role of the *interviewer*. It is his/her obligation to prepare (see Preparation) and conduct (see Section Application) the interview, and to process the information gathered (see Section Result processing). An interviewer should be: *knowledgeable*, *structuring*, *clear*, *gentle*, *sensitive*, *open*, *steering*, *critical*, *remembering* and *interpreting* (based on [Kvale2008], explained in detail in Section Application).

The stakeholder is the *interviewee*. He/she answers the questions asked by the interviewer, thus expressing their requirements.

It is the *note-taker's* job to record all relevant information provided by the interviewee. He/she needs to know in advance what the elicitation objective of the interview is and must know and understand the interview guide. Only if the note-taker knows the context of the interview and the topic (including terminology) sufficiently, is he/she able to follow the interview and decide which information to take down and which not.

The interviewer may also take over the role of note-taker. If possible, however, this should be avoided, as it requires the interviewer to jump back and forth between two different cognitive modes, requiring additional energy and usually negatively affecting the performance in both roles. The role of note-taker may be replaced by a sound or video recording device. The advantages and disadvantages are discussed in the Section Preparation.

Preparation

Define the elicitation objective(s) and the required result quality as part of defining the elicitation activity (see 1.3.1).

Select a suitable stakeholder(s) for the interview who you expect will be able to answer your questions in order to reach your elicitation objective(s), while achieving the required result quality. You will never know for sure in advance whether you have selected the right stakeholders. With good stakeholder management and properly researched stakeholder documentation, however, you will significantly increase the chance of making a good choice.

Prepare an interview guide. There are many different ways to do this: some people prefer a list of bullet points, showing the intended order and hierarchy of questions; others prefer a guide in mind-map style, where the questions are ordered clockwise around the interview objective. The latter form can make it easier to follow the interviewee's trail of thought when he or she is jumping from one topic to another while answering the question that was originally posed.

The interviewer develops an overall structure for the interview, reflected by the interview guide. After deciding which questions to ask and which questions have what priority, thought should be put into how to do introductions, the purpose of the interview, what the kick-off question should be, the order of the questions and how the interview should end. The guide does not have to contain the exact wording of the questions to be asked (at least in a qualitative interview). Nonetheless, in order to be *clear* in their questioning, interviewers should think about how they are going to phrase their questions.

Another organizational matter is that of time and place. The interviewer must set a suitable time that fits the interviewee's schedule. It is advisable to add another 15 minutes to the planned interviewing time to allow for warm-up time and as a general buffer. The interview itself should not exceed 60 minutes. For most interviewers and interviewees 20-40 minutes will be a good time frame.

The interview should take place in a separate room, allowing for concentration and confidentiality. Ideally, the location should be close to and convenient for the interviewee.

The interviewer must decide in advance how the interview should be recorded. There are three options: the interviewer takes notes him-/herself, a note-taker takes notes, or an audio or video recording.

These are the advantages and disadvantages of the three options:

	Interviewer takes notes	Note-taker takes notes	Audio/video recording
Advantages	• No additional person or device is required. • While taking down notes, interviewer and interviewee gain time to think.	• The interviewer can concentrate fully on posing questions and listening to the answers/observing the interviewee. • Interviewer and note-taker have two perspectives on the same event (possibility to uncover cognitive biases).	• The interviewer can concentrate fully on posing questions and listening to the answers/observing the interviewee. • All information is available unfiltered after the interview.
Disadvantages	• Some information may get lost or distorted. • The interviewer has to switch between two cognitive modes (requires energy). • During note-taking, long awkward pauses may arise • Notes are usually very short and may not be understood anymore after the interview.	• Some information may get lost or distorted. • If the note-taker is not fully involved in the interview topic he/she might misunderstand information and take down wrong or incomplete information.	• Information filtering starts only after the interview; the whole interview has to be listened to, again. • Legal restrictions may apply, depending on national law or company regulations. • The interviewee might prohibit the recording.

Hint 3.1.1:

Often options 1 and 3 are combined, i.e. the interview is recorded and the interviewer still takes some notes (but he or she can focus on asking the questions and listening to the answers).

If recording of the interview is planned, it is advisable to get approval from the interviewee and his/her organization in advance.

Application

During the interview, the interviewer leads the interviewee by asking questions. The interviewer puts into practice the following qualifications [Kvale2008]:

- *Knowledgeable*: has an extensive knowledge of the topic; knows what issues are important to pursue
- *Structuring*: prepares and follows a structure for the interview. Communicates the structure to the interviewee during the interview
- *Clear*: uses simple language, and poses clear, simple, easy and short questions

- *Gentle*: lets people finish; gives them time to think; tolerates pauses
- *Sensitive*: listens attentively to what is said and how it is said; is empathetic in dealing with the interviewee
- *Open*: responds to what is important to interviewee; is open for new aspects introduced by the interviewee and follows them up
- *Steering:* knows what he/she wants to find out. The interviewer controls the course of the interview and is not afraid of interrupting digressions.
- *Critical*: is prepared to challenge what is said, for example, dealing with inconsistencies in interviewees' replies
- *Remembering:* can recall earlier statements and relates what is said to what has been said earlier in the interview
- *Interpreting*: clarifies and extends the meanings of the interviewee's statements; provides interpretations of what is said, which may be confirmed or negated by the interviewee

The stakeholder(s) (interviewee) answer(s) these questions and the Requirements Engineer (interviewer) listens carefully, checking for several things, e.g.:

- Whether the stakeholder has understood the question and is providing the desired information
- Whether the Requirements Engineer understands what the stakeholder is saying
- Whether the question is fully answered
- Whether the stakeholder is sending relevant non-verbal information
- Whether the note-taker is taking down the required information

The note-taker listens to the interviewer and interviewee, filters the relevant information from the conversation and takes it down. He/she watches the interviewer closely to catch non-verbal or verbal signals on what to take down.

[Portigal2013] and [BaCC2015] provide more insights into the art of interviewing, focusing on user interviews.

After the interview, the *interviewer* and/or *note-taker* prepares the interview notes and sends them to the interviewee(s) for review. This serves two purposes: Firstly, you make sure you have understood all information from the interview correctly and have not forgotten any important aspect. Secondly, you show appreciation for the interviewee's time and input.

Hint 3.1.2:

Set a date until when you require the interviewee's feedback.

It is also advisable to send a short and friendly reminder a few days before the deadline expires.

Result processing

After the interviews, the collected (raw) data needs to be analyzed and aggregated into useful information. This could be requirements, needs, goals, problems, user groups, scenarios, processes, artefacts, etc.

An affinity diagram [BaCC2015] may help in processing the collected data. See the following example that illustrates the two main aspects of an affinity diagram:

1. Extract insights from your collected data and write each insight on a (green) card.
2. Clustering: Group your insights/cards into clusters and label each group (yellow cards).

Figure 18: Affinity Diagram

Typical artifacts

During preparation, execution and post-processing of an interview, the following artifacts are usually created or updated:

- Interview guide
- Notes and/or audio/video recordings taken during the interview
- Final notes of the interview (to be sent to all participants).

Opportunities

The big advantage of an interview is direct feedback - not only verbally but also non-verbally. By watching your interviewee closely, you can pick up much more information than just what he or she is saying. You can also react to any information you receive right away.

Also, an interview can be realized at quite short notice. Although it should never be unprepared, an experienced interviewer may only need about an hour in advance to prepare the interview sufficiently.

Challenges

Conducting and particularly evaluating interviews is very time-consuming. If the Requirements Engineer is missing some of the qualifications mentioned, an interview can easily get out of hand, and not deliver the intended results. Asking the wrong questions, or the right questions in the wrong way, may drastically reduce the interview's usefulness.

Variants

There are several different interview variants, e.g.:

- **Open/qualitative interview:**
  The interview does not follow a rigid structure. The interviewer may deviate from his/her initially prepared topics or questions during the interview. If several interviews are conducted on the same topic, each interview has its individual course.
  This variant is pretty common in requirements elicitation.

- **Partly standardized interview:**
  Some questions are standardized and have to be asked in the exact same way in every interview. The rest of the interview is an open interview.
  The standardized part is used to gather quantitative data.
- **Fully standardized/quantitative interview:**
  The order of the questions, their exact wording and their possible answers are given. It aims to provide comparable results. They can be processed directly with statistical methods.
  In Requirements Engineering this variant is not very common. It is mainly used in market research.
- **Group interview:**
  Several interviewees are asked by one interviewer within one interview session.
  This variant has the advantage that differences of opinion between the interviewees can be addressed right away. This type of interview is, however, not advisable if there are personal differences between the interviewees, or if one holds a more senior position than the other. Both factors are likely to influence the outcome of the interview negatively.

3.1.1.2 Questionnaire

What is a questionnaire?

With a questionnaire, several people are asked to answer, in writing, the same set of questions, which are presented in a structured way. There are two main kinds of questionnaires: *quantitative* and *qualitative*.

Quantitative questionnaires are used to confirm hypotheses or previously elicited requirements. They can be evaluated quickly and deliver statistical information.
Quantitative questionnaires use closed-ended questions (see 3.1.1).

Qualitative questionnaires are suited to the elicitation of new requirements. They tend to deliver complex results and thus are usually far more time-consuming to prepare and to evaluate.
Qualitative questionnaires use open-ended questions (see 3.1.1).

Quantitative questionnaires can also include some open-ended questions and qualitative questionnaires may include some closed-ended questions.

Role of participants

In a survey by questionnaires[5], two roles are required: questionnaire writer and respondents.

The Requirements Engineer takes the role of *questionnaire writer*. It is his obligation to design and distribute the questionnaire and to evaluate the results.

The *respondents*, who are stakeholders, receive the questionnaire and are asked to respond to it by a specific date.

Depending on how the questionnaire is issued, other roles may be involved, such as a data processor to enter data from paper questionnaires into a data processing system or to process the data manually.

[5] The term “questionnaire” describes the tool which is then used in a survey by questionnaires.

Preparation

As a survey by questionnaires cannot be corrected once the questionnaires have been distributed, you should invest sufficient time and thoughts into the preparation phase.

First of all, you should clarify whether there are any legal or company-specific restrictions applicable to your project concerning how a survey is conducted. In some companies, surveys have to be approved by a specific council.

Define the elicitation objective(s) and the required result quality as part of defining the elicitation activity (see 1.3.1).

Select suitable stakeholders for the survey. How many participants are needed for a valuable or representative result? A quantitative questionnaire needs to deliver representative results. For a qualitative questionnaire representative results may not be the focus, but valuable ones, delivering new requirements.
When selecting the participants, you should consult the stakeholder list. If the participants are not on the list, or contact information is missing, update the stakeholder list.

Set the maximum length of the questionnaire: the longer the questionnaire, the fewer the responses. How many questions are acceptable for your participants?

Select the form of presentation. Can you use a survey tool, or do you have to use paper?

Formulate the questions with regard to the elicitation objective and target stakeholders. Use open-ended questions to elicit new aspects and requirements; use closed-ended questions to verify requirements you have already elicited, or to confirm or reject a hypothesis based on the requirements elicited to date.

If using closed-ended questions, decide on a suitable type of scale. Also make sure you are not using many different kinds of scales throughout your questionnaire. Check for all closed-ended questions whether you need to provide a “not applicable” option. If such an option is missing – especially when questions are obligatory – this may falsify the result, as the participant is forced to answer the question even if he or she cannot answer it.

Define the order of the questions, and make sure that order is logical. Take considerable care that the answers to some questions do not exclude other questions (e.g. “Do you drink wine?” and “Do you prefer red wine or white wine?” If the participant answers the first question with “no”, the latter question does not make any sense).

Plan the timeframe for your survey. When will you send out the questionnaire? How much time should you allow for responding? When should you send a friendly reminder? How much time will you need for result evaluation?

Test your questionnaire before you distribute it. Have it reviewed by others and have it answered by a few test candidates in order to identify unclear questions. As soon as the questionnaire is distributed, you can no longer correct any flaws. Also, execute a trial evaluation of the data provided by your returned test questionnaires. Often you only find out when doing the evaluation that you missed an important aspect, or that the questions do not deliver the exact data that you wanted to probe. Furthermore, having done the evaluation once helps you to do it efficiently again with the data from the real questionnaire.

Application

Distribute the questionnaire to the participants. When sending the questionnaire via email, take into consideration whether the recipients should be disclosed or undisclosed. Usually, undisclosed recipients are used.

In your announcement with the distribution, communicate the context of the questionnaire: Who are you and why are you sending them a questionnaire? What is the purpose of the survey? How long will it take to answer the questionnaire? What will happen with the results?

Will the participants be informed about the results? If yes, by when? In order to increase the number of potential responses, it might be a good idea to announce an incentive.

Result processing

For qualitative questionnaires, result processing will usually take quite some time. The more responses you get, the more time is required for evaluation. Here, you will usually have to deal with conflicting, or potentially conflicting, statements and different levels of requirements.

Hint 3.1.3:

An affinity diagram [BaCC2015] may help in processing the collected data (see the example of an affinity diagram in chapter 3.1.1.1 on processing data collected from interviews).

For quantitative questionnaires, result processing can be done automatically (if the survey is carried out electronically). Using a Web-based survey tool, the number of responses does not influence the amount of time needed for aggregating the data. However, the intellectual interpretation of the statistics may still take some time.

Typical artifacts

During preparation, execution, and result processing of a survey by questionnaires, the following artifacts are usually created or updated:

- Questionnaire
- Processed results (e.g. statistics, diagrams, report)

Opportunities

A survey by questionnaire is an asynchronous elicitation technique, meaning that the Requirements Engineer and the stakeholder do not have to be at the same place at the same time to use this technique.

With a questionnaire, it is possible to involve many stakeholders at once, either for collecting new requirements from many stakeholders at the same time or for evaluating existing requirements with a large set of stakeholders and in this way confirm or disprove a hypothesis (e.g. “The users are happy with the system as is”).

Challenges

Using a questionnaire survey can be very time-consuming: “One of the biggest misconceptions about a survey is the speed with which you can prepare for, collect and analyze the results. A survey can be an extremely valuable method, but it takes time to do it correctly.” [BaCC2015]

Especially in the case of qualitative questionnaires can the evaluation become extremely time-consuming.

A major drawback of questionnaires is the limited (or non-existent!) possibility of feedback in case of doubts. This applies for both the respondents as well as for the Requirements Engineer. Thus, questions as well as responses may be interpreted incorrectly.

[BaCC2015] also describe the following points to be aware of:

- *Selection Bias*: Often out of convenience, respondents are only selected who are easy to come by (e.g. own department, friends and family). Such a group is not representative, which may result in inaccurate data.
- *Non-response Bias*: Even if the stakeholders to whom the questionnaire is distributed are selected representatively, an imbalance may occur as some of them will answer and some of them will not. The response rate may range between 20% and 60%.

- *Satisficing*: If the questionnaire requires too much cognitive effort they may apply satisficing, a strategy to reach satisfactory but not ideal results. E.g. respondents may select the same choice for all questions.

Variants

Apart from the previously discussed variants of qualitative and quantitative questionnaires, there are also the following variants:

- **Paper-and-pencil questionnaire:**
  The questionnaire is answered on paper. This results in a high processing effort as data has to be computerized prior to evaluation. For this type of questionnaire, the effort for evaluations rises with the number of responses, for both qualitative and quantitative questionnaires.
- **Computerized/Web-based questionnaire:**
  The questionnaire is answered online. For quantitative questionnaires the responses can be aggregated automatically.

3.1.1.3 Requirements workshops

Workshop is an umbrella term for group-oriented techniques. They can be conducted in very different ways, include other elicitation techniques or even process patterns (e.g. a Design Thinking workshop within an agile development). Workshop formats differ from small informal meetings to organized events with several dozen or even hundreds of stakeholders.

What is a requirements workshop?

In this handbook we focus on the requirements workshop as defined by [Gottesdiener2002]:

"A requirements workshop is a structured meeting in which a carefully selected group of stakeholders and content experts work together to define, create, refine, and reach a closure on deliverables (such as models and documents) that represent user requirements."

A requirements workshop differs from any other meeting in two main ways:

- It is a facilitated meeting (i.e. planned, structured and moderated).
- It is focused on the elicitation[6] of requirements.

Role of participants

In a requirements workshop there are four distinguishable roles: *facilitator*, *participant*, *recorder* and *workshop sponsor*.

The Requirements Engineer takes the role of *facilitator*. It is his/her obligation to prepare (see Section Preparation) and facilitate the requirements workshop (see Section Application), to make sure all relevant information from the workshop is recorded (see Section Application) and to derive requirements from the information gathered during the requirements workshop (see Section Result processing).

The facilitator should be neutral to the outcome and does not need to be a content expert. It might even be a hindrance if the facilitator is a content expert because there is a risk that he or she has a (hidden) stake in the outcome.

[6] Such workshops often also cover conflict resolution issues. This, however, is not the focus of this chapter.

The *participants* are stakeholders. It is their task to create the workshop products, based on the workshop objective and guided by the facilitator.

The *recorder* records the group's work. This could be in writing and/or by taking pictures or videos. Depending on the complexity of the requirements workshop, this role may also be filled by the Requirements Engineer together with the role of facilitator. The recorder should also not be a content expert. Although he/she does need to understand enough of the topic discussed to be able to record the information correctly.

The *workshop sponsor* authorizes and legitimizes the workshop, including the workshop objectives and result quality. He/she is responsible that all the required resources (i.e. people and money) are available.

Preparation

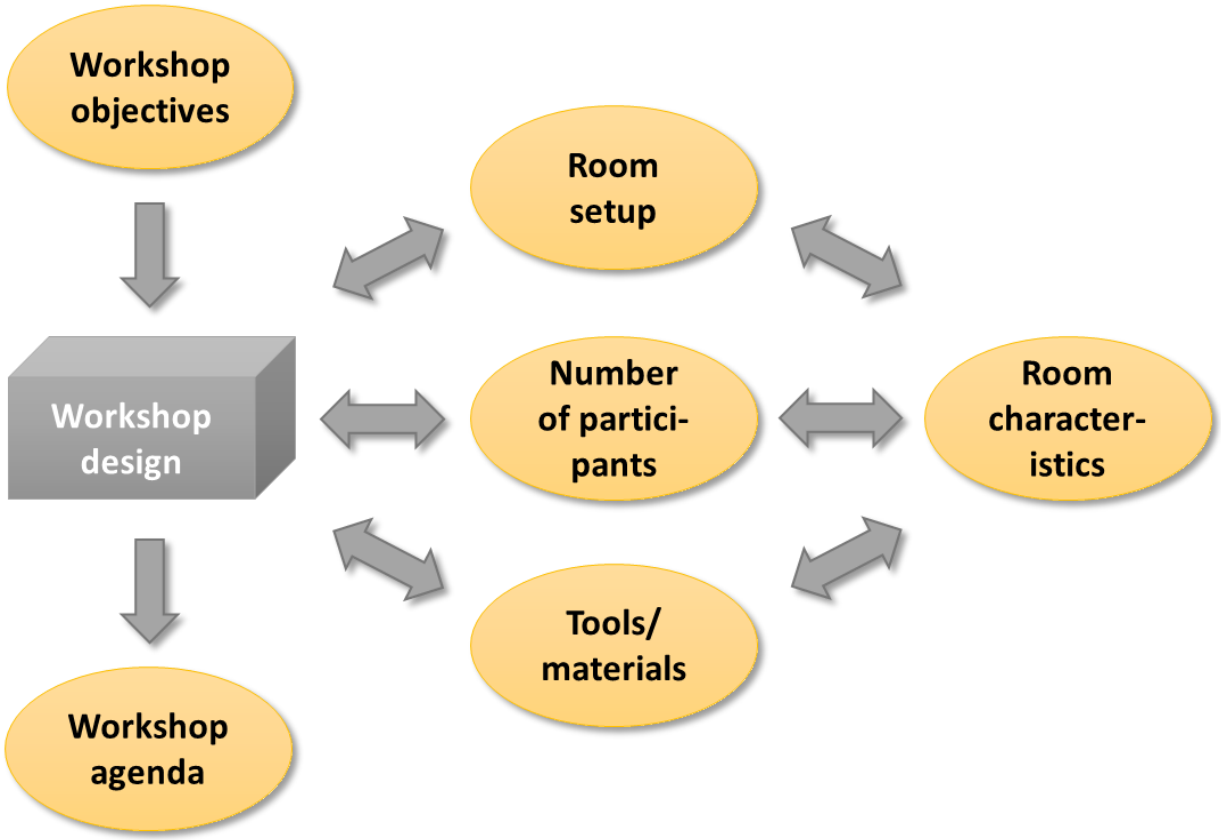

Figure 19: Typical Interdependencies of Workshop Preparation Activities

A *workshop design* influences and is influenced by several factors (see Figure 19). As some of those factors are both influencing and influenced, creating a workshop design is an iterative process.

Define the *workshop objectives* (elicitation objectives) and the required result quality as part of defining the elicitation activity. Both topics have to be clarified with the workshop sponsor.

The room setup depends on the *workshop design,* while conversely the workshop design depends on the *room setup*. Whereas the room setup and the *room characteristics* influence each other.

In an ideal world, the workshop design would be independent from the room characteristics. In most cases, however, the room is either chosen in advance or the room selection is restricted by distance and/or budget. Hence, in most cases, the workshop has to be set-up according to the room characteristics, resulting in the room setup: Which material (e.g. flipcharts, pinboards, whiteboards, posters) will be placed where in the room? Are tables required? How should they be placed? Are chairs required? How should they be placed? Where will the workshop results be stored?

Room characteristics and *number of participants* are also interdependent: the more participants, the bigger the room, or the smaller the room the less participants. The workshop design also depends on the number of participants. Some workshop models are not as scalable as others. The ideal number of participants for a requirements workshop is 7-12 participants. Whereas the maximum of participants should be 16 unless there is more than one facilitator [Gottesdiener2002].

As with the room characteristics, *tools and materials* should not be influencing factors on the workshop design, but in many cases they in fact are. They are usually also influenced by the room characteristics: for example, if the room does not have a huge whiteboard to work with, it cannot be used as a tool during the workshop and other tools have to be used, such as post-it notes.

When these interdependencies have stabilized, the participants can be invited. The facilitator needs to clarify with the workshop sponsor who is going to be invited to the workshop. Also clarify with the sponsor whether he/she will be present. In general, the workshop sponsor should be present for opening and possibly also for closing the workshop. During the workshop the presence of the workshop sponsor usually is not required, and in some cases is also not recommended (e.g. if the sponsor's hierarchical position might interfere with the workshop results). Communicate time and place as well as the workshop objectives in the invitation. Ideally you can already provide an initial *workshop agenda*.

The *workshop agenda* is the result of the workshop design. It is the guideline for the workshop. A workshop agenda should allow enough buffers such that timings can be adjusted during the workshop. Some workshop activities may need considerably more time than planned, while others may be quicker than anticipated.

Application

As the detailed application is highly dependent on the workshop design, only general advice is given in this Section.

Before the participants arrive, the room has to be prepared (e.g. put up posters, move tables and chairs, etc.). Also make sure that refreshments are available.

Welcome the participants as they arrive.

Open the workshop and explain the workshop objectives and the expected output (this may also be done by the workshop sponsor).

Set ground rules for the workshop, ideally in cooperation with the participants.

Lead the group through the workshop. Make sure all participants are involved and all input delivered by the participants is recorded. Keep checking whether any of the following have to be adjusted: workshop activities, time frame, tools used, room climate.

Close the workshop: Thank the participants for their input. Clarify next steps and responsibilities. Inform them what is going to happen with the workshop results.

Result processing

After the workshop, the facilitator or recorder process the workshop documentation and make it available to the participants.

As workshop results, just like workshop setups, are highly diverse, no general guidelines for result processing can be given.

Typical artifacts

Artifacts depend on the type of workshop (see Section Variants), workshop design and type of recording.

Typically, variants of the following artifacts are generated:

- Workshop agenda
- Workshop documentation (e.g. pictures, flipcharts, digital notes, minutes)

Opportunities

Requirements workshops support team communication and decision making. They bring together different stakeholders to achieve a better mutual understanding of the project or product. Potential conflicts and misunderstandings can be identified and solved during requirements workshops that might not become obvious with other elicitation techniques.

Challenges

As requirements workshops require a lot of resources (people, time and money), they can hardly be realized without a committed sponsor. The time needed for workshop preparation and result processing is often underestimated. The workshop results depend highly on the workshop design. Inexperienced facilitators tend to underestimate the effort required and are often lacking the flexibility and toolset to prepare and hold an effective workshop, as well as to process the results efficiently.

Hint 3.1.4:

If new in the role of facilitator, you should work together with a more experienced facilitator in all phases of the workshop.

Variants

There are many different variants of requirements workshops. They can be combined with or take the form of other well-known workshop models such as:

- **World café** [BrIs2005]:
  A setup of several tables. Each table is dedicated to one topic and has one moderator. The group at a table (4-6 people) discusses that topic for 15-30 minutes. Then the teams are mixed and take their places at another table.
- **Open space technology** [Owen2008]:
  The participants suggest topics for discussion. Several topics are discussed at the same time in different locations (rooms). The topic owner is responsible for documentation of the discussion. The participants can freely join and leave a discussion (law of two feet). At the end of the open space the groups come together and the topic owners present the results to the whole group.
- **Design Thinking**:
  See pattern description in Section 1.4.4.

3.1.2 Observation techniques

The aim of observation techniques is to extract requirements from observation of, e.g. processes, users, or typical usage situations.

Special attention should be given to the investigators' simplification bias [BaCC2015]: Inexperienced observers (i.e. novices to the domain) have the tendency to oversimplify the expert user's problem-solving strategies while observing them. Thus, it is highly recommended to acquire knowledge about the topic (e.g. talk to a subject matter expert) before using observation techniques to minimize this bias. It is also advisable to let subject matter experts review the observation notes afterwards.

In practice, the three techniques presented in this handbook: field observation, apprenticing and contextual inquiry, may overlap. It is therefore important to do the transition from one technique to the other, or to create a conscious mix of techniques. What should be avoided is drifting from one technique to another e.g. a contextual inquiry to a pure field observation, or vice versa. If there are reasons to mix the techniques, mix them, but keep in mind what is important for each element.

3.1.2.1 Field observation

The Requirements Engineer observes the stakeholders during their work in their usual environment without interfering. The observations made are used to derive requirements which have to be confirmed by review or further elicitation techniques. Field observation is also known as job shadowing, following people around, observation, or pure observation.

What is field observation?

During field observation, the Requirements Engineer watches the stakeholder (usually the end user) in his/her environment while he or she is performing the tasks for which a system is planned to be developed or improved. The important difference between field observation and apprenticing or contextual inquiry is that no interaction takes place between the observer and the observed subject(s).

Field observation is typically used in situations where interaction with users is not possible (e.g. it would be a distraction) or would interfere with the process itself and potentially falsify results. It can also be applied in public places without even informing the observed subjects (e.g. sit with other patients in a doctor's office and observe them during waiting times).

Role of participants

The Requirements Engineer takes the role of *observer*. The stakeholder is the o*bserved subject*. There may be more than one observer as well as more than one observed subject.

Preparation

Define the elicitation objective(s) and the required result quality as part of defining the elicitation activity and select a suitable stakeholder(s) for the observation.

Here, the elicitation objective is especially important, as it helps you to focus on the relevant aspects during the observation. As our brain is not able to watch everything all the time, it is of utmost importance to know exactly what you are looking for.

For field observation, time and place have a major influence on the result. Advance research may be required to find out the right time and place.

Some influencing factors for time and place may be:

- The process to be observed only takes place at certain times (e.g. people starting their work day, food distribution in a canteen).
- The process takes place in different locations at different times (e.g. people getting on or off public transportation).
- Parts of the overall process to be observed take place at different times and/or different places (e.g. a part is produced in one production line and then handed over to another production line where it is not processed right away).

Decide in advance whether or not the observed subjects will be informed that they are being observed. In any situation where it is obvious that you are observing (e.g. you observe someone who usually works alone in a room), you should inform the observed subject in advance. In other cases, it may not even be possible to inform the observation subjects (e.g. observation in a public place like a supermarket).

Application

When the observed subject is informed about the observation it may be a good idea to brief them on why you are doing it and that they should try to forget about your presence and do their work as usual.

During field observation, the observer involves all their senses (seeing, hearing, smelling, feeling) to gather information about the observed situation/process(es) and takes notes on all information that may be relevant with respect to the elicitation objective. Depending on the situation, it may be possible to take pictures and/or videos.

As mentioned in the preparation Section, it is very important to have your elicitation objective in mind during the observation. However, there is also the risk of being too focused and therefore biased during the observation. Figuratively speaking, the elicitation objective is your lighthouse: it helps you navigate through the observation and decide whether something is relevant or not for reaching the elicitation objective.

Do not participate in or interfere with the situation(s) being observed.

In some cases, it may be possible to perform the field observation via camera (e.g. previously installed security cameras), thus avoiding interference with the process by physical presence.

[BaCC2015] suggests considering the following points during field observation:

- What language and terminology do people use?
- If you are observing the use of an existing system, how much of the system/software/features do users actually use?
- What barriers or stop points do people encounter?
- If what you are interested in is task-focused:
    - How much time do people devote to accomplishing a task?
    - What questions do people have to ask to accomplish a task?
    - What tools do users interact with as they are trying to accomplish a task?

Result processing

Remark: result processing in general is identical for field observation and contextual inquiry.

Each field observation session leads to a large amount of collected raw data. This needs to be analyzed and aggregated. Possible supporting techniques for result processing of a field observation could be:

- User-Task-System-Context-Issue analysis, i.e. apply perspective-based reading to extract the information on these five aspects from the collected raw data. The result is an intermediate artifact on which the subsequent analyses can be made. These analyses should be carried out in the team, preferably involving stakeholders and technical team members.
- Alternatively, the collected material can be processed with an affinity diagram. [BaCC2015]
- From the raw data collected during field observation, or based on the intermediate artifacts created according to the previous two points, identify and document:
    - Real user groups or personas
    - Processes and frictions in these processes
    - Cultural model, physical model, and flow model [BeHo1998]
    - List of important artifacts used during task completion
    - List of issues
    - Goals, problems, needs, and requirements

Typical artifacts

During preparation, execution and post-processing of a field observation activity, the following artifacts are usually created or updated:

- Notes from the observation (e.g. text, drawings)
- Artifacts collected during the observation (e.g. pictures taken, videos)
- Processed results, e.g. as proposed by [BeHo1998]:
  - Sketches of interesting factual connections
  - Physical model that shows the context of work
  - Cultural model showing the cooperation between participating users
  - Flow model showing process interruptions

Opportunities

Field observation is a very valuable technique if interaction with users is not possible or not intended [BaCC2015]. It helps in gathering information that cannot be expressed verbally by the stakeholders (e.g. implicit knowledge) and helps to understand the stakeholder's situation and thus be more empathic in subsequent interactions.

A huge advantage of field observation is that it does not require additional stakeholder resources: the stakeholders perform their tasks as usual.

According to [Koelsch2016], field observation also may help to

- identify work or process flows,
- identify what things bother the user,
- notice any awkward steps they encounter, and
- identify any room for improvement.

Challenges

The observer's bias influences what he or she actually sees. He/she may think they understand what a certain action is about but may be totally wrong. Field observation should never be used on its own, but should always be followed by additional elicitation techniques (e.g. interview).

Depending on what is being observed and where it is being observed, field observation can be very time consuming, especially when there are long periods where nothing relevant happens.

Variants

[BaCC2015] also describe "deep hanging out" as a more structured form of observation. The observer becomes a user themselves (e.g. using public transportation), and it uses a formal structure to organize the observation process, for example by focusing on these ten focal points: family and kids, food and drinks, built environment, possessions, media consumption, tools and technology, demographics, traffic, information and communication access and overall experience.

3.1.2.2 Apprenticing

We make here a strict differentiation between field observation and apprenticing: field observation is non-participatory, while apprenticing is participatory. In the literature there are differing definitions where both concepts, or even all three concepts, including contextual inquiry, are called apprenticing (see [RoRo2013]).

What is apprenticing?

Apprenticing follows the idea of masters and apprentices [RoRo2013]. The Requirements Engineer does a short internship in the environment in which the system to be developed/improved later will be used (or is already in use).

Experienced subject matter experts ("masters") teach the Requirements Engineer ("apprentice") in order to empower him/her to better understand the domain and therefore to better elicit requirements. "Whatever the work, it seems sensible to have a fair understanding of it before attempting any changes" [RoRo2013].

An optimal duration for the internship depends on many different factors, e.g. complexity of the process, high repetitiveness vs. low repetitiveness, time availability of master and apprentice.

The important difference between apprenticing and contextual inquiry is that in apprenticing the Requirements Engineer actually practices the job under investigation and learns about context and tasks not just by observing and asking but primarily by doing – which by nature takes more time than performing a contextual inquiry. The latter is just about observation of the expert doing a task in his context and verbal cognitive task analysis together with the expert (no practical doing by the Requirements Engineer).

Role of participants

The Requirements Engineer becomes the apprentice. It is his/her task to learn from the stakeholder, who becomes the master.

Preparation

Define the elicitation objective(s) and the required result quality as part of defining the elicitation activity and select a suitable stakeholder(s) as masters for the apprenticing (for details, see below).

As with field observation, the elicitation objective is especially important for apprenticing, as it helps you to focus on the relevant aspects during the apprenticing.

The Requirements Engineer should collect any information in advance that might help him or her to become a better student in the apprenticing situation, and thus avoiding the need for basic questions during the apprenticing. This way, the time spent by the stakeholder is used efficiently.

From an organizational point of view, a suitable master has to be found and they have to be willing to participate in the apprenticing. If the master is unmotivated to do the apprenticing it is best to switch to a different elicitation technique. He/she has an active and essential part in this technique. We suggest a preparatory meeting with the master to clarify how and when the apprenticing is to take place.

It is a good idea to instruct the master to be suspicious if the apprentice remains quiet for a long period. He/she should then ask questions like "what are you thinking about?" or "tell me what's on your mind".

Application

The master teaches the apprentice the relevant processes and or actions. The main target of apprenticing is to experience the context in which the later solution is to be applied. The Requirements Engineer should therefore perform at least some of the steps in the overall process themselves. The Requirements Engineer avoids making assumptions but voices everything they conclude, thus enabling the master to affirm or correct those conclusions.

It is important to ask a lot of questions during the apprenticing to get as much information as possible and to avoid false assumptions.

The apprentice takes notes during the apprenticing.

In some situations, it may be possible for the apprentice to perform the task or tasks on his/her own, if the master considers it appropriate.

Result processing

Right after the apprenticing, the Requirements Engineer should write down as many aspects as possible from the apprenticing, to preserve the knowledge that has been gained.

Then, the data gathered from the apprenticing is analyzed.

The following questions may be used for guidance:

- What difficulties have I experienced?
- What could be made easier?
- What did I consider complicated?
- What would be the biggest benefit?

Typical artifacts

During preparation, execution and post-processing of an apprenticing activity, the following artifacts are usually created or updated:

- Notes from the apprenticing (generated during and afterwards)
- Artifacts collected during the apprenticing (e.g. screenshots taken, training material)
- Processed results (should be reviewed by the master!)

Opportunities

Apprenticing helps in understanding the system environment and application domain. The Requirements Engineer experiences the difficulties and workarounds used instead of just learning about them. Interaction with stakeholders during the apprenticing usually improves the relationship with them, which later improves the communication with these stakeholders. Also, as the Requirements Engineer has experienced the stakeholders' work environment, he/she might be considered as part of the group and less of an outsider. It also helps the Requirements Engineer to understand the stakeholder's situation and thus be more empathic in subsequent interactions.

With apprenticing, implicit knowledge can be elicited and turned into explicit knowledge.

Challenges

The quality of an apprenticing rises and falls with the stakeholder's motivation and didactical capabilities. Thus, it is important to clarify those aspects prior to the apprenticing. A considerable investment is required from the stakeholder: during the apprenticing they cannot get their work done, or at least not at their usual pace.

Variants

If the Requirements Engineer does not actively do the tasks but only follows the stakeholder for a long time, asks and gets things explained by the stakeholder, this is usually called a "job shadowing".

Note:

Job shadowing for a short period of time is rather a variant of contextual inquiry.

3.1.2.3 Contextual inquiry

What is contextual inquiry?

Contextual inquiry is an iterative, field data-gathering technique, where the Requirements Engineer studies a few carefully selected users in depth to arrive at a fuller understanding of the work practice across the entire user base [BeHo1998]. It is a rich combination of observation and discussion. The Requirements Engineer (and other accompanying team members) observe the users in their own work context. Furthermore, the Requirements Engineer discusses with the users their tasks and engages with them to uncover unarticulated aspects of their work. The goal is to discover structures and patterns, and to find out how the user organizes his work.

Contextual inquiry is based on four principles:

- *Context*: Go to the user's own context to observe them performing their tasks.
- *Partnership:* Try to get into a relationship that is a partnership:
  Together, try to understand the user's work practice.
- *Interpretation:* Develop a shared understanding with the user about the aspects of work that matter.
- *Focus:* In the preparation of the contextual inquiry, define elicitation objectives and direct your investigation to gather the relevant data in order to reach the objectives.

The main differences between apprenticing and contextual inquiry are that apprenticing usually takes much more time (usually 1-3 days) and that the Requirements Engineer in the role of the apprentice is actually practicing the relevant tasks, whereas a contextual inquiry session usually takes 60-90 minutes and the Requirements Engineer just observes and talks to the stakeholder who is performing the task.

Role of participants

The user is the domain expert. The Requirements Engineer is the expert for recognizing work structures, patterns and differences in individuals' work organization. In this inquiry session both are partners, trying to unveil relevant aspects of the user's work together.

Other participants take the role of observer. Each of them may focus on different aspects, e.g. used artefacts, physical context, communication, process, culture, etc.

Preparation

The preparation of a contextual inquiry depends on the target group. Where access to users is limited or potentially costly (e.g. stock exchange traders, top managers, users abroad, etc.), the effort for preparation may be many times that of the actual execution of the inquiry. In such cases, the Requirements Engineer's understanding of what he/she is going to see must be established in advance (e.g. by interviewing a retired person from the target group or by asking supporting staff in the context, attending user training, watching videos, and studying tutorials or other written material). If the target group is easily accessible, the basic understanding can be gained through a single interview with someone who knows the users, and then the first contextual inquiry can focus on really getting to know the tasks and context of the user.

These are the general tasks for the preparation of a contextual inquiry:

- Depending on the elicitation objective, determine the number, type and location of the users to be visited.
- Arrange appointments with the selected users and ask them to prepare typical – or specific – tasks which they will work on during the contextual inquiry.

- If you plan to do a series of contextual inquiries, iteratively prepare each individual inquiry session:
    - Who is going to participate (if possible, involve stakeholders and/or other team members to let them also experience users in the field)?
    - Depending on the elicitation objective: What shall be the focus of this inquiry? Who is focusing on what aspect?
    - How to record the findings (prepare checklists for your notes; check whether it is permissible to take audio or video recordings, photos, etc.)?

Application

When arriving on-site, introduce the participating individuals and your elicitation objective. Explain the partnership setting: the user is the expert for the task at hand; the Requirements Engineer is the expert for recognizing work structures, patterns and differences in people's work organization. Then ask the user to perform the prepared tasks and to explain what he/she is doing and why. If you have permission, do not forget to start the recording.

Observe the user during the contextual inquiry, take photographs (if allowed), collect artifacts that the user works with, make notes and ask the user regularly about aspects of how he or she performs the tasks. In this way, the user is forced to make things explicit that allows the Requirements Engineer to uncover the attitudes, goals, problems and needs of the user. Furthermore, the team can get information about the real processes and frictions in the workflow, cultural aspects of the user and his/her co-workers, information flow, physical environment, etc.

After the contextual inquiry, the Requirements Engineers sometimes conducts an interview with the user to go through a list of prepared questions. However, this would be an insertion of another elicitation technique. Conclude the contextual inquiry with a de-briefing.

Result processing

Remark: result processing in general is similar for field observation and contextual inquiry.

Each contextual inquiry session leads to a large amount of collected raw data. This needs to be analyzed and aggregated. Possible supporting techniques for result processing of a contextual inquiry could be:

- User-Task-System-Context-Issue analysis, i.e. apply perspective-based reading to extract the information on these five aspects from the collected raw data. The result is an intermediate artifact on which subsequent analyses can be based. These analyses should be carried out in the team, preferably involving stakeholders and technical team members.
- Alternatively, the collected material can be processed into an affinity diagram. [BaCC2015]
- From the raw data collected during the inquiry session, or based on the intermediate artifacts created according to the previous two points, identify and document:
    - Real user groups or personas
    - Processes and frictions in these processes
    - Cultural model, physical model, and flow model [BeHo1998]
    - List of important artifacts used during task completion
    - List of issues
    - Goals, problems, needs, and requirements

Repeat these steps for each contextual inquiry you perform, thereby systematically enriching your understanding of the aforementioned aspects (users, tasks, systems, context, and issues).

Typical artifacts

During preparation, execution and post-processing of a contextual inquiry, the following artifacts are usually created or updated:

- Notes and audio recordings from the contextual inquiry
- Artifacts collected during the contextual inquiry (e.g. pictures taken, videos, completed forms, notes, lists, tools, etc.)
- Processed results, e.g. as proposed by [BeHo1998]:
    - Sketches of interesting factual connections
    - Physical model that shows the context of work
    - Cultural model showing the cooperation between participating users
    - Flow model showing process interruptions

Opportunities

While a challenge with field observation is avoiding misinterpretations of the observed user behavior, it is a key feature of contextual inquiry to address and verify such observations immediately in the context. Furthermore, users do not have to think about how to suitably explain their way of working in a meeting room presentation. Instead, they just perform their job (tasks they usually do every day) and explain to the Requirements Engineer what they are currently doing, thus reflecting their own course of action and making their expert knowledge explicit. The Requirements Engineer and the user discuss, on the basis of a work task just carried out, problems, factual relationships, motivations, and opportunities for improvement.

Contextual inquiry is the supreme discipline for human-centered requirements elicitation [RiFl2014]. In a short period of time, the Requirements Engineer can effectively find out a lot of information about the users, their goals and tasks, the systems used, the relevant context and real problems and issues.

Challenges

Depending on factors such as the Requirements Engineer's existing knowledge of the domain, the availability of suitable users for carrying out the contextual inquiry, the geographical location of the context to be visited and the number of potential user groups, the number of contextual inquiries to be planned may be high and the corresponding preparation and execution effort may be considerable.

Be sure to select the correct users. Since contextual inquiry is a qualitative elicitation method, the distribution of selected users does not necessarily have to be representative in a statistical sense, but nevertheless they should represent the expected user groups.

Although the Requirements Engineer can steer the contextual inquiry and motivate the user to participate, a contextual inquiry is only possible as long as the user truly participates in it. With an unmotivated user, contextual inquiry will not lead to satisfactory results.

Contextual inquiry is not just a simple combination of observation and interview. The preparation and implementation of a contextual inquiry is completely different and requires appropriate training and experience.

Variants

The condensed ethnographic interview [DeDe2011] is characterized as a “top-down” – in contrast to contextual inquiry’s “bottom-up” – approach, because you start with a semi-structured interview, which is followed by an observation of how users accomplish their tasks in the real context, focusing on processes and tools. Artifacts are also collected and discussed.

3.1.3 Artifact-based techniques

Artifacts are products of human work, such as IT systems, documents, images, audio and video files, etc. Some types of artifacts are relevant as sources of requirements. It is usually a time-consuming task to examine artifacts in detail - especially if those artifacts contain too much (irrelevant) information or are poorly structured. Nonetheless, artifact-based techniques are very potent and provide significant benefits, particularly when stakeholders are not readily available.

In this Section we cover four artifact-based techniques: perspective-based reading, system archaeology, reuse of requirements and crowd-based Requirements Engineering. Perspective-based reading is the main technique and a basic skill for the other three techniques. There is also some overlap between the techniques. Thus, it is important to understand the essence of each technique.

3.1.3.1 Perspective-based reading

What is perspective-based reading?

Perspective-based reading was first investigated as a review technique by [BaGL1996]. They discovered that reviewers found more defects in less time using perspective-based reading, as compared to their usual way of reviewing requirements documents. As a review technique, each user takes the perspective of a particular user of the requirements document (e.g. developer, tester or architect).

Since then this technique has also proven valuable as a requirements elicitation technique when dealing with documents as requirements sources.

Role of participants

The only role required for perspective-based reading is that of the reader.

Preparation

Define the elicitation objective(s) and result quality as part of defining the elicitation activity (see 1.3.1).

Define the source document and select the appropriate perspective(s) for the source document.

Which perspectives are suitable depends on the source document and the elicitation objective? Suitable perspectives could be: requirements for specific functionality, requirements on usability, functional requirements, non-functional requirements, etc.

Clarify how many readers (Requirements Engineers) will take part in perspective-based reading.

Application

One reader can only apply one perspective in one reading session. If more than one perspective is relevant for a source document, the Requirements Engineer has to take these perspectives one after the other.

If there are more readers, each reader covers a different perspective. The readers may read through the documents in parallel (if it is a digital document or if several analog copies are available), even in the same room, or asynchronously. If the latter, a target date should be agreed.

When reading through the source document, each reader scans the document for content relevant for their current perspective and then reads through any potentially interesting Section in more detail. As soon as they find information that is relevant from their perspective, they record it in an appropriate way.

Example:

The source document is a competitor system's user manual. One reader takes the perspective "quality requirements", while another reader takes the perspective "stunning features". Both scan through the user manual, looking for relevant information from their perspective. When they find a potentially interesting section in the manual, they read through it in detail, extracting the requirements. The first reader might stop at any sentence containing numbers (indicators for potential non-functional requirements); the second reader might stop at certain chapters where system functionality is described.

Result processing

Passages identified as potentially relevant for the system to be developed must be verified by application of other elicitation techniques or by a review of the requirements. We strongly suggest combining perspective-based reading with stakeholder-focused techniques (e.g. interview, contextual inquiry).

Typical artifacts

During preparation, execution and post-processing of perspective-based reading, the following artifacts are usually created or updated:

- Notes from the original investigation for suitable documents
- Potentially: Physical or digital copies of the document(s) with markings and/or notes for relevant Sections
- Documentation of extracted information (reading notes)
- Processed results

Opportunities

When stakeholders are not readily available, perspective-based reading is a good way of eliciting potential requirements. Furthermore, already documented knowledge is reused and does not get lost.

Challenges

It is often hard to tell whether a document is still up to date and valid. When using obsolete documents for perspective-based reading, usable results may be few for the time invested.

As with all artifact-based techniques, the Requirements Engineer never knows which requirements elicited from documentation remain relevant for the system to be developed. Further techniques are needed.

Variants

Not applicable.

3.1.3.2 System archaeology

What is system archaeology?

In system archaeology, requirements are extracted from existing systems: legacy systems as well as competitor systems or even analogous systems (systems in a different context with similar functionality). We will call these *source systems* in the following discussion.

This technique is mainly used if an existing system has been used (and possibly changed) over many years and is now to be replaced by a new system, because, for example, the legacy system's technology is no longer compatible with its neighboring systems, its performance no longer meets requirements or new functionality is required.

Role of participants

System archaeology does not involve specific roles.

Preparation

Define the elicitation objective(s) and result quality as part of defining the elicitation activity (see Section 1.3.1). The elicitation objective might also give you a hint of where to look: if, for example, the elicitation objective is to elicit the requirements for the system's data, the system's database might be a good place to start.

Select the source system from which you would like to derive requirements. In most cases, that would be a system in use (legacy system) that is about to be replaced by the system to be developed. However, competitor systems or analogous systems to the system to be developed may also be sources for system archaeology.

Next, collect all potentially relevant documents for the source system, e.g. user manual, test cases, architecture documentation and project charters. If you can get your hand on requirements documentation, use the technique reuse of requirements (Section 3.1.3.3).

There may be cases where no current, useful documentation for the source system is available, or where the documentation alone is insufficient for system archaeology.

In such cases you may need the source system executable on a suitable environment (usually a test environment, where data can be manipulated without using live data) and/or the source code.

If you need to extract information from the source code and are not trained in the programming language used, you will need someone to assist you.

Application

The application differs depending on whether documentation is the source for system archaeology or if it is source code or the system itself.

Documentation

If your source is documentation, such as user manuals or test cases, apply perspective-based reading (Section 3.1.3.1).

Source code

Depending on the programming language used, different reading strategies for the source code apply (e.g. assembler vs. object-oriented programming language). Also, well-documented code will be much easier to read than undocumented code.

Generally, it is useful to find out how the code is structured and which naming conventions are used, as well as familiarizing oneself with the programming style used, before going into detail.

Usually, source code analysis is only required to find out specific details in the implementation.

Example:

The source system is executable on a test environment and you have been able to extract all the main features and functionalities. However, there is one feature where you cannot simulate a specific behavior in the test environment as you do not have sufficient test data. So, you search for this specific feature in the source code and analyze the rules implemented for it.

Executable source system (UI analysis)

If the source system is available in an executable form, the system can be analyzed in a structured way. Good practices are:

- Follow the business processes implemented in the system:
  - Follow one business case after the other.
  - Look out for exceptions and branches.
  - Often helpful: Document the knowledge acquired using models (e.g. using BPMN, UML).
- Follow the UI structure:
  - Analyze one screen after the other, starting with the start screen.
  - Analyze element after element, following the writing direction of the language used (e.g. left to right and top to bottom for English).
  - Document the screen flow and button functionality with screen shots and/or storyboards or other suitable methods.

Result processing

The requirements identified as potentially relevant for the system to be developed must be verified by application of other elicitation techniques or by review of the requirements. We strongly suggest combining system archaeology with stakeholder-focused techniques (e.g. interview, contextual inquiry).

Typical artifacts

During preparation, execution and post-processing of system archaeology, the following artifacts are usually created or updated:

- Notes from the original investigation for suitable source systems and their documentation
- Potentially: Physical or digital copies of the document(s) with markings and/or notes on relevant Sections
- Documentation of extracted information (reading notes, including models or drawings)
- Processed results (e.g. model-based documentation of identified processes)

Opportunities

System archaeology ensures that no requirements implemented in the source system get lost. It is especially useful if no other current documentation is available to find out what the system *really* does. Additionally, it is not dependent on stakeholder availability.

Challenges

System archaeology is potentially very time-consuming, as many documents or lines of code have to be checked for relevance and read. A particular pitfall is that this technique helps only to uncover existing functionality, and does not say whether that functionality is still needed or even correct. In the worst-case errors from the old system may be implemented again in the new system!

Variants

Not applicable.

3.1.3.3 Reuse of requirements

What is reuse of requirements?

Although every project is by definition unique and, as such, results in a unique product, there is a good chance in every project that there is potential for reuse, also of requirements (e.g. roles and profiles or login procedures).

Conscious reuse avoids reinventing the wheel over and over again.

There are different forms of requirements reuse. In the case of product lines, a high percentage of requirements can usually be reused, either as they are or with some modification.

Example:

The Flux compensators of the DeLorean time machine version 1 and DeLorean time machine version 2 will differ in some specific features, however the overall functionality will be the same.

Even if two systems seem at first sight to be totally different, they may still have similarities in requirements.

Example:

The time- and place-setting device and the heating system of the same time machine have completely different functionality. But as they are both part of the same system (the time machine), they share a high percentage of non-functional requirements (e.g. maximum acceleration).

Even systems that have no interface to each other and are not part of a common super-system may still have similar requirements, as the processes they cover have similar steps.

Example:

An airplane and a time machine have something in common in that they both move people. Some requirements concerning life support systems (e.g. oxygen supply) or safety requirements will probably be the same for both.

Reuse of requirements has three aspects: the elicitation aspect, the documentation aspect and the requirements management aspect. The elicitation aspect covers the general principle of reuse and its role in gathering requirements for a new project. The documentation aspect addresses the question of how to document requirements to support later reuse. The requirements management aspect addresses the question of where and how to organize requirements to support later reuse.

In this handbook we only cover the first aspect.

Hint 3.1.5:

If your product will be available in several variants, or if there will be several releases, invest some thought and time into setting up the documentation structure and using a requirements management tool to make it easier to reuse your elicited requirements.

Role of participants

Requirements reuse does not involve specific roles.

Preparation

Define the elicitation objective(s) and result quality as part of defining the elicitation activity (see Section 1.3.1).

Select the documents that might contain requirements for reuse. Typical documents are requirements specifications of interfacing systems or of previous versions of the system to be developed. While the reuse of functional requirements is typically only possible between similar systems, non-functional requirements can also be reused between systems that do not seem to have many similarities at first sight. This makes non-functional requirements especially interesting for requirements reuse.

[RoRo2013] suggest searching for documents with potentially reusable content by walking through their *Volere* template, [Robles2012], Chapter 15:

1. The Purpose of the Project: Are there other projects in the organization that are compatible or that cover substantially the same domains or work areas?
2. The Client, the Customer, and other Stakeholders: Can you reuse an existing list of stakeholders, a stakeholder map, or a stakeholder analysis spreadsheet? Users of the Product: Do other products involve the same users and thus have similar usability requirements?
3. Mandated Constraints: Have your constraints already been specified for another project? Are there any organization-wide constraints that also apply to your project?
4. Naming Conventions and Definitions: You can almost certainly make use of parts of an existing glossary.
5. Relevant Facts and Assumptions: Pay attention to relevant facts from recent projects. Do other projects' assumptions apply to your project?
6. The Scope of the Work: Your project has a very good chance of being an adjacent system to other projects that are underway in your organization. Make use of the interfaces that have been established by other work context models. Consider your work scope and ask whether other projects have already defined similar business events.
7. Business Data Model and Data Dictionary: Are there business data models from overlapping or connected projects that you could use as a starting point?

Application

Search the identified documents for requirements that might be relevant for your current product to be developed. This is an application of perspective-based reading (compare Section 3.1.3.3). [RoRo2013] suggest focusing on the verbs to find reusable requirements, as they represent processes.

Only some reusable requirements will present themselves on a silver platter. To find the others, you will have to interpret and abstract what you are reading.

Result processing

The requirements identified as potentially suitable for reuse must be verified by application of other elicitation techniques or by review of the requirements. We strongly suggest combining reuse of requirements with stakeholder-focused techniques (e.g. interview, contextual inquiry).

Typical artifacts

During preparation, execution and post-processing of reuse of requirements, the following artifacts are usually created or updated:

- Notes from the investigation for suitable documents
- Potentially: Physical or digital copies of the original requirements documents with markings and/or notes on reusable Sections

- Processed results, e.g. abstract models of existing specifications (see [RoRo2013])

Opportunities

Reuse of requirements avoids creating anew what already exists. Furthermore, it helps not to forget important requirements. If the requirements specifications used are well-structured and well-written, a great many requirements can be elicited in a short time.

Challenges

If the requirements specifications used are neither well-structured, well-written or up to date, reuse of requirements can become a very time-consuming endeavor and may not deliver the desired result.

Another potential danger is copying wrong requirements that do not apply to the new system, or indeed that were already wrong in the original document. Additionally, by copying existing solutions, new innovation may be suppressed.

Variants

Not applicable.

3.1.3.4 Crowd-based Requirements Engineering

What is Crowd-based Requirements Engineering?

Crowd-based Requirements Engineering (CrowdRE) is also known as crowd-centric Requirements Engineering (CCRE).

We define CrowdRE according to [Groen et al.2017] as "an umbrella term for automated or semi-automated approaches to gather and analyze information from a crowd to derive validated user requirements."

Such information could be feedback given in a forum, ratings and reviews in app stores, usage data, and others. Text and usage mining are typical techniques used in CrowdRE.

Role of participants

The main participant in CrowdRE is "the crowd", usually consisting of existing or future users of the product to be developed. However, it is not only restricted to users [LiFi2012].

Such a crowd may already exist, or is "created" in the course of CrowdRE, e.g. by creating a gamification experience that forms the crowd [Snijders et al.2015].

The Requirements Engineer does not have a dominant role in this elicitation technique. He or she gathers or initiates the techniques for gathering data from the crowd and evaluates the results. In some forms of CrowdRE, it may not even be apparent for the crowd members that they are participating in a requirements elicitation process (e.g. reviews in app stores).

As the application of CrowdRE may require specifically developed software (e.g. gamified elicitation platform, usage data analysis tool, feedback module within product to be developed), CrowdRE may also require a development team to create such software.

Preparation

As CrowdRE can take very different forms, there is no standard way of preparation.

In general, it can be said that the right crowd for the product to be developed has to be defined as well as an appropriate way to involve the crowd and evaluate the elicitation results – which are usually huge amounts of raw data (big data). [Snijders et al.2015] and [LiFi2012] suggest concrete methods/tools.

Application

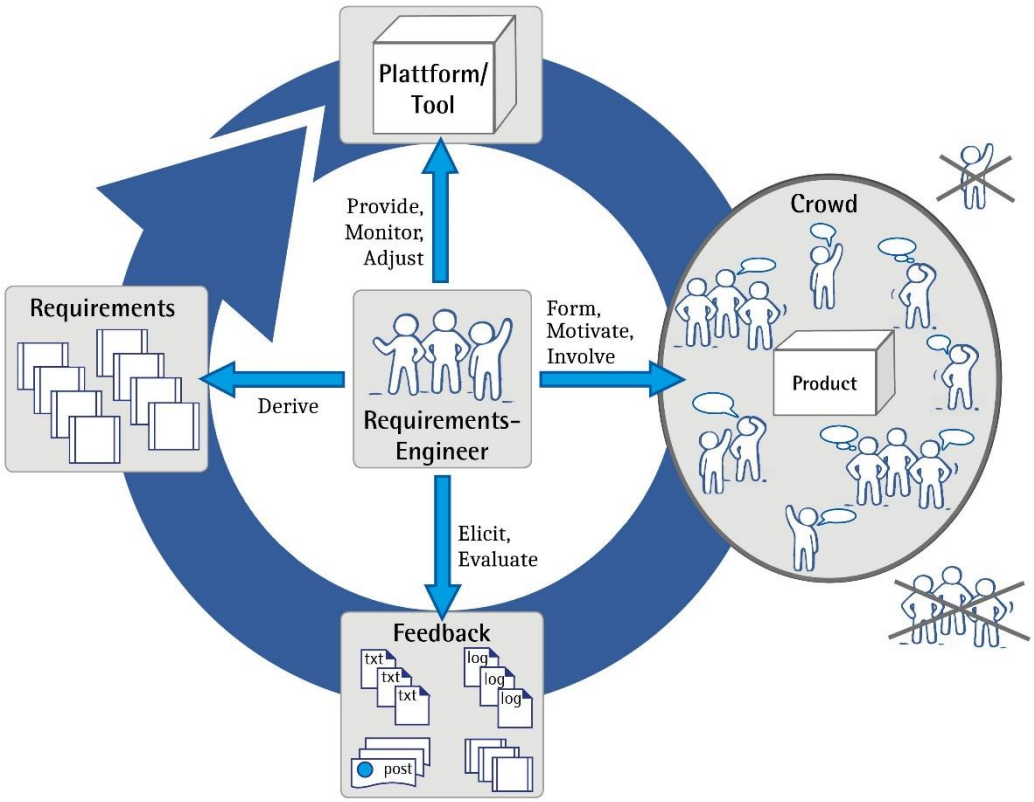

Figure 20: The relationships among the aspects of crowd-based Requirements Engineering according to [Groen et al.2017]

According to [Groen et al.2017], the Requirements Engineer needs to motivate the crowd members to provide user feedback on the product (see Figure 20). That feedback is elicited and analyzed by the Requirements Engineer (e.g. using text mining tools). In addition, he or she analyzes contextual and usage data. The derived requirements are implemented by the development team. The new product release is handed back to the crowd to provide feedback and validate the requirements elicited.

Result processing

As with preparation, no typical result processing can be described. Usually, tools and methods to deal with big data have to be applied (e.g. text mining).

Typical artifacts

Not applicable.

Opportunities

CrowdRE provides the chance to collect representative data from actual users. With the right tools and a suitable set-up, CrowdRE has the potential to reduce the overall cost of elicitation over time. Customer-bonding can be improved and their creativity can be used directly by involving them actively. Furthermore, requirements cannot only be elicited but also prioritized and validated by the crowd.

Challenges

CrowdRE is usually quite time-consuming to set up. Even if data is used that is already "there" (e.g. ratings in an app store), suitable ways of evaluating the data have to be found and applied.

If specific platforms are provided to encourage CrowdRE, these have to be developed or bought, and administrated. If no crowd can be motivated to take part in CrowdRE, the results may be worthless. If there is a very active crowd, on the other hand, huge amounts of data are created that have to be evaluated to derive requirements.

Another potential pitfall of CrowdRE is the creation of a biased view as only the feedback of people who participate in crowd communication is acquired. Also, CrowdRE is vulnerable to manipulation and pranks (e.g. in a crowdsourcing competition by Henkel to find a new package design for a dishwasher detergent a cover with the slogan "Schmeckt lecker nach Hähnchen" (tastes yummy like chicken) got the most votes [Katie2017]).

Variants

Crowd Testing is a related field in which the crowd is instrumentalized to provide valuable feedback. When identifying a (usability) defect, participants of a crowd testing activity often are invited to provide suggestions for a better solution. In such a case it might be valuable to co-operate with the testers to gather defects and requirements from the same crowd basis.

3.2 Design and idea-generating techniques (L2)

The traditional task of Requirements Engineering was to gather and document the necessary requirements from all relevant stakeholders for a subsequent development process (cf. [Boehm2006]). This task led to the application of the gathering techniques already introduced above (see Section 3.1). The growing influence of software as an innovation driver in many businesses eventually led to a new understanding of Requirements Engineering as a creative, problem-solving activity (cf. [Couger1996] or [Maiden et al.2010]).

Outside the software and Requirements Engineering community, the broader term design techniques has emerged. Design techniques comprise creativity techniques for idea generation and provide additional or combined techniques to elaborate ideas to gain further insights for a given idea [Koes1964] A. Koestler: The Act of Creation. Last Century Media, 2014.

[Kumar2013]. Popular design techniques include prototyping (e.g. mock-ups), storyboarding and scenarios.

In the following discussion, we present two creativity techniques (brainstorming and analogy technique) and two design techniques (prototyping and scenarios/storyboards) as examples of techniques that are applicable in various contexts. However, many more techniques can be found in the literature (cf., e.g. [Koes1964] A. Koestler: The Act of Creation. Last Century Media, 2014.

[Kumar2013]). The discipline of design also provides several principles that support, on the one hand, the development process of systems (e.g. process concepts), and, on the other hand, the development of the system itself (e.g. form, shape or functionality of the system). Presenting and discussing these principles is beyond the scope of this handbook. A good overview of design principles is presented, e.g., in [LiHB2003].

Preconditions for creativity

Although there are many techniques that aim to generate creative results, none of them guarantees success. Several mechanisms in our brain have to come together to enable creative ideas. Simplified, the following four preconditions have to be met for creativity to emerge:

- ***Chance*** – and therefore time – for an idea to come up:
  An idea is a new connection between neurons. Whether such a connection comes into existence or not is mainly a matter of chance. It can be improved, however, by positively influencing the other three preconditions for creativity and, especially, by giving it time.

- ***Knowledge*** of the subject matter, which raises the odds for an idea that makes the difference:
  Knowledge is the soil from which creative ideas can sprout. By acquiring knowledge on a certain topic, the respective regions in our brain are activated, improving the chance of new neuronal connections. This means that if we want to have new, innovative ideas on a specific topic, we have to first gain knowledge on this topic and then think about the topic in order to activate the respective regions in our brain. This doesn't mean that only subject matter experts can come up with innovation in their field of knowledge; outsiders might introduce fresh, unconventional ideas – **if** they learn enough about the topic in focus (e.g. a specific step in a process which should be improved or functionality and constraints of an existing product that needs a face-lift).
- ***Motivation***, as our brain can only be creative if there is a direct benefit for its owner:
  Being creative is an energy-consuming task for our brain. Our brain's prime directive is: save energy. As such, it will only invest the energy required if it anticipates a benefit. That could be a raise in status (i.e. better chance to survive) or as simple as having fun.
- ***Safety and security***, as useless ideas must not have negative consequences:
  If we fear that a useless idea threatens our status or safety in any way, our brain will shut down any creativity and "go blank". It switches into safety mode and sticks with the known and safe ways of doing things.

With the application of any design or idea-generating technique, it is necessary to make sure the four preconditions for creativity are met for the participants.

3.2.1 Brainstorming

What is brainstorming?

Brainstorming was developed by Alex F. Osborn [Osborn1948] as a group creativity technique to support the development of new ideas for a given question or problem. As with most creativity techniques, the crucial point of brainstorming is to defer judgment by separating the finding of ideas from the analysis of ideas. A good overview of brainstorming in RE is provided by [MaGi2001] and [Pohl2010].

Role of participants

Participants of a brainstorming session develop the ideas. A moderator takes care of the brainstorming rules. A note taker can be used to support the moderator in visualizing the developed ideas.

Preparation

Before the brainstorming takes place, the lead question or problem has to be defined. The elicitation objective (see syllabus EU1) can be used as a starting point or even directly as a lead question. As a second step, the participants of the brainstorming session have to be identified. As a rule of thumb, the participants should come from various areas. A deep understanding of the question is not necessary for all session participants. Having some participants from different domains can be an advantage, because they can offer completely different perspectives on the question.

Application

At the beginning of a brainstorming session, the moderator explains the lead question and the brainstorming rules to all participants. [Pohl2010] describes (based on Osborn) these seven rules for brainstorming in Requirements Engineering:

- Quantity over quality
- Free association and visionary thinking are explicitly desired.
- Taking on and combining expressed ideas is allowed and desired.
- Criticizing other participants' ideas is forbidden even if an idea seems to be absurd.
- Questions for clarification are allowed.
- Even at longer-lasting deadlocks do not abort immediately – overcome at least two longer-lasting deadlocks.
- Wait until the brainstorming comes to a natural end.

The moderator starts the session and all participants can express their ideas for the lead question. The moderator (or an additional note taker) visualizes the ideas for all participants. During a brainstorming session, the flow of ideas will usually stop after a certain time. Such breaks do not indicate the end of the brainstorming. A new statement from one participant can restart the flow of ideas immediately. The group should overcome at least two such breaks before the session comes to a natural end (see rules above).

Result processing

After the brainstorming session has taken place, the list of ideas has to be analyzed. Each idea typically consists of a short statement. Therefore, additional effort has to be spent to detail the best ideas, for example by performing an additional workshop with some of the brainstorming participants. Usually, prioritization techniques are applied while processing brainstorming results.

Typical artifacts

The outcome of a brainstorming session is the list of produced ideas. If all brainstorming participants agree, an audio or video recording of the brainstorming can be an additional output for later analysis.

Chances

Brainstorming is useful for developing a large number of ideas in a short period of time. Ideas that might be considered absurd in the beginning may inspire the group to develop other unexpected and innovative ideas. Supporting this group dynamic effect (i.e. ideas mutually inspiring each other) is one major benefit of a good brainstorming session.

Challenges

Brainstorming results are only rough sketches of ideas. The Requirements Engineering work starts after the brainstorming session. The developed ideas have to be evaluated, prioritized and further detailed.

The group dynamics effects are the main driver for developing good ideas. Initiating and maintaining a creative atmosphere in the brainstorming session is therefore the main challenge.

Variants

Many different variants of brainstorming have evolved over time, for example:

- Brainstorming paradox follows the same procedure as a normal brainstorming. The difference is in the brainstorming topic, which is the opposite of the normal brainstorming topic. The aim is to explore risks and perils in relation to a topic (e.g. "What factors help to make potential buyers angry and make them leave our web shop?").

3.2.2 Analogy technique

What is the analogy technique?

The analogy technique (cf., e.g. [Robertson2001]) is a technique that helps to come up with ideas for critical and also complex topics. It uses analogies to support thinking and generating ideas. Its success or failure is mainly influenced by the quality of the analogy. Good sources for analogies are related systems (see Chapter 2).

Role of participants

The Requirements Engineer takes the role of *moderator* who develops the analogy and presents it to the group.

Participants of the analogy technique elaborate the given analogy and transfer the developed ideas to the original problem.

Preparation

The preparation for the analogy technique consists of two steps.

Select a proper analogy for the given problem. The selected analogy can be close (e.g. the same problem in another domain) or distant from the original problem. For example, the original problem is to improve the buying process of life insurance products over the internet. A close analogy could be a buying process of a different product over the Internet (e.g. buying a book instead of an insurance policy in an online store). A distant analogy could be a situation in which a waiter advises the guest on menu selection (the menu is the analogy for the insurance policy).

Select the participants for the analogy technique. The participants should be experts in the original problem. If possible, experts (or at least knowledgeable persons) in the selected analogy should be invited. It is not necessary that one participant is an expert both in the original problem and the analogy. It is also possible to select the analogy together with the participants.

Application

The application of the analogy technique consists of two steps:

- Elaborate the selected analogy. The participants elaborate positive and negative aspects of the analogy in detail (e.g. by using whiteboards or a flip chart). This step should be performed without referring to the original problem.
- Transfer to original problem. Once the analogy has been elaborated, the participants examine every aspect of the analogy and transfer these to the original problem.

Result processing

The participants create a list of statements for the original problem that have been derived from the analogy. To achieve a final result, the participants have to examine each statement and discuss the value of the statement for the given problem. During this discussion, the participants can further elaborate the transferred statement to improve its value for the given problem.

Similar to the brainstorming technique, the outcome of the analogy technique requires further effort in order to analyze and prioritize the results.

Typical artefacts

The outcome of the analogy technique is a detailed description of the analogy, the list of transferred statements for the original problem and the final list of evaluated and detailed statements for the original problem.

Opportunities

A well selected analogy allows the development of innovative ideas for a given problem. Especially if the given problem is hidden from the participants, the analogy technique may come up with unexpected and unconventional ideas.

If the given problem is difficult and/or sensitive, the analogy technique can help to create an open and constructive working environment. The challenging part is of course the transfer of the findings to the original problem.

Challenges

The main success factor for the analogy technique is the selection of a proper analogy. If the participants are aware of the original problem, it is important to keep the given problem out of the discussion of the analogy.

Variants

Bisociation [Koes1964], derived from the terms "bi" (two completely different things) and "association", lets the participants of a creativity workshop associate ideas for a given problem statement with something that seems to have nothing in common with the problem (e.g. 5 pictures of famous painters, some special physical objects, short video clips of interesting animals).

3.2.3 Prototyping

What is prototyping?

A prototype is an umbrella term for any intermediate artifact that is created to investigate certain characteristics or alternative solutions for a system to be developed by means of some kind of tangible experience. In most cases, it relates to characteristics that cannot easily be understood or defined upfront in models or described in documentation. Prototypes can range from very simple paper sketches or clickable user interface simulations to physical instances of a device or initial implementations of software; they possess specifically chosen – but not all – characteristics of the future system, and allow for the investigation of certain other, as yet unclear, characteristics.

It is the use of the artifact for investigation, elaboration, clarification, design, testing, validation etc. (see [LiHB2003]) that defines it as a prototype. In short, the purpose of prototyping is to experience (certain parts of) systems "… before they are real, even before we have arrived at their final design, much less their implementation" [Buxton2007].

T.Z. Warfel [Warfel2009] describes eight guiding principles for the use of prototyping:

- Understand your audience and intent
- Plan a little - prototype the rest
- Set expectations
- You can sketch
- It's a prototype - not the Mona Lisa
- If you can't make it, fake it
- Prototype only what you need
- Reduce risk - early and often

Role of participants

In Requirements Engineering, prototyping is used for both elicitation and validation. In elicitation, participants usually have an internal role, such as Requirements Engineers, digital designers, UX designers, or other members of a development team.

Their primary goal is to understand the problem, find appropriate solutions (in a creative process) and eventually come up with new requirements or refine existing ones. In validation, participants often have an external role: they may be users or other stakeholders outside of the development team who evaluate the implementation of certain requirements. This, too, can lead to new requirements and the refinement of existing ones.

In both situations, three major roles can be distinguished:

- *Moderator*: Usually this is the Requirements Engineer, who decides to use prototyping as an elicitation technique, plans and manages the prototyping, instructs the other participants, analyses the results and draws the conclusions.
- *Developer*: A prototype has to be designed, built and tested before it can be used. For simple prototypes like sketches, this role can be fulfilled by the Requirements Engineer, but more sophisticated prototypes may require the support of designers, developers, or tool specialists.
- *Investigator*: Investigation of the prototype is often a team effort, in which, depending on size and complexity, Requirements Engineers, designers, programmers, testers, key users and other stakeholders may be involved. To focus the investigation on the desired characteristics, it is important to follow the guidelines or scripts from the moderator, but also to allow some time for free exploration.

Preparation

As prototypes can be very different in nature, they all require a different approach in their preparation [McElroy2017]. For a simple sketch, pencil and paper may be sufficient; for a physical prototype, a 3D-printer and extensive software might be necessary.

The one – and most important – thing in common is a plan: the Requirements Engineer must have a sound idea of what characteristics to investigate and how to do it. It must also be clear what other, already established characteristics (not subject to investigation) should be in place and which aspects shall be deliberately omitted. For instance, if you want to prototype a web shop to analyze performance, the functionality itself should be working.

It is good practice to test the prototype for the established characteristics, as defects in these areas may have a negative impact on the investigation of the characteristics that are in focus. Also, it must be clearly communicated which parts or functionalities of the system are out of scope (i.e. not available or not working in the prototype), especially when external participants are involved.

Depending on the size and complexity of the prototyping, preparation may include developing guidelines, instructions, scripts, procedures and templates for the investigators (see, for instance, preparation for a usability test [UXQB2017]).

Application

The most common applications for prototyping include [Warfel2009]:

- **Shared communication**
  This is using prototyping as a collaborative tool, a lingua franca (common language) among all participants. The goal is to create a shared understanding of requirements amongst the business, Requirements Engineers, designers, developers and users. It often involves simple sketching to determine the outline of the future system and may introduce a number of variants from which to choose for further development.

- **Working through a design**
  Prototypes are a great way to actually work through a design, test it out, see which alternatives will work, and flesh out the details. It mostly relates to elaboration and refinement of earlier high-level requirements and helps to maintain integrity and consistency.
- **Selling your idea internally**
  "Showing is better than telling." By creating a quick prototype of different design options, it is easier to convince business stakeholders of the benefits of your design, to make the right choice and to gain support for it.
- **Usability testing**
  Usability and other quality characteristics are notoriously difficult to catch in specifications and models, but easy to experience in a prototype, even early in a project.
- **Gauging technical feasibility and value**
  For every design of a new system, the ultimate questions remain: "Can it be built?" and "Will it add value?". Building and operating a prototype can generate trust by providing positive answers to these questions, or – if negative – prevent costly project failures at an early stage.

Result processing

Collect all the findings of the prototyping and decide what to do with them in the team together with key stakeholders (e.g., in a walkthrough). Some feedback needs to be reviewed with a larger group of stakeholders; some improvement ideas need to be checked with the technical staff.

Subsequent activities depend on the project situation: for example, the development of a new prototype to explore other characteristics of the system, the choice of a preferred variant for further development or the detailed specification of the discovered requirements for a chosen design.

Typical artifacts

- The prototype
- Guidelines, instructions, etc.
- Analysis and documentation of the findings and conclusions, such as ideas, new requirements, clarified existing requirements for future reference

Opportunities

Prototyping is a versatile tool that can be applied in various stages of Requirements Engineering. Early stages benefit from exploring ideas, later stages benefit by elaborating and refining requirements:

- Prototyping allows Requirements Engineers to get qualitative feedback from stakeholders early in the project and thus to avoid costly defects later.
- Prototyping helps Requirements Engineers to think about the problem, to oversee and reduce complexity and thus to focus on delivering value to the customer.
- Prototyping can be used as an iterative, participatory, joint design approach that allows the development team and stakeholders to elaborate solutions together.
- Prototyping fits perfectly into modern development approaches, like Agile, DevOps, Lean and Design Thinking.

Challenges

Depending on the type of prototype selected, its development may require significant effort. For example, a realistic, clickable functional model of a user interface (horizontal prototype) or a physical prototype may take several days for design, development and testing. The challenge is to balance the effort of creating the prototype with the expected benefits from the obtained insights. Remember the saying "A prototype is not the Mona Lisa" and resist the temptation to make the prototype a little bit more perfect.

Another challenge (especially for sophisticated prototypes) is that the prototype could be considered the final product by some stakeholders. Such a misconception may lead to unrealistic expectations on the project schedule as the stakeholders often underestimate the time needed to transform a prototype into production-ready software.

Variants

While prototypes can vary broadly both in intention and functionality, three main characteristics are usually discernible: fidelity, lifecycle and notion.

Fidelity is about the congruence of the prototype with the target product:

- A *low-fidelity (LoFi) prototype* resembles the future system just enough to allow for some relevant experience with it as far as the intended characteristics are concerned, e.g., screen sketches on a storyboard.
- A *high-fidelity (HiFi) prototype* mimics the external interfaces of the future system to a high degree, so that at first sight one can hardly see the difference (once again as far as the intended characteristics are concerned). Internally, 'underwater', the system may be far from complete.

Lifecycle is the relationship of a prototype to the target product:

- An *exploratory prototype* is built only for the purpose of investigation and evaluation. It is also called a "*throwaway" prototype*, because it will be discarded after use. This kind of prototype is common in mass production.
- An *evolutionary prototype* will be continuously elaborated, extended, improved and refactored, until it ends up as the final product. During its iterative development it is first used for investigation and then refactored, based on the findings. This kind of prototyping is often encountered in one-off production. One could argue that Agile/Scrum development is usually a kind of evolutionary prototyping.

Notion, i.e. prototypes can be seen from different viewpoints during development:

- *Feedback* may be the prime purpose, for instance when a Requirements Engineer wants to evaluate a certain aspect of the solution, such as the user interface, with a broader group of stakeholders.
- *Design* may come first. This is the case when a prototype is created to explore and compare different solutions for a certain problem, as is often done in industrial design projects.

3.2.4 Scenarios and storyboards

What is a scenario?

The word scenario is derived from the Latin word "scaenarium", which means "place for erecting stages". Nowadays, the word scenario is used to refer to an outline or a synopsis of a play[7]. In the same sense, scenarios can be used to create an outline of the usage of a system [Carrol2003].

Scenarios can be documented in a written or a visual form. The visual form of a scenario is called a storyboard. A storyboard typically is a series of panels that show sketches of a scene or action in a series of shots (as for a film, TV show or commercial)[8]. Again, in the same sense, a storyboard can be used to describe a flow of actions for a system.

Preparation

Preparation for the scenario technique consists of a decision regarding the scenario story or stories, and collecting material and information for the development of the scenarios. The story of the scenario refers to the concrete action or actions that should be described. Remember, scenarios come from theater and a play always has a story. The story can be derived from the goals or problems that have to be solved. Be aware, defining the scope of the story is a challenge.

If the story is too big (e.g., an end-to-end business process in an online shop), the scenario may become too long. Long scenarios are like overly long books: they are not really readable. Conversely, making the scope too small is also a risk as the scenario becomes too fine-grained and one might get lost in detail (for example, adding one product to a shopping cart might be too narrow for a scenario). As a rule of thumb, one should try to identify a scope that describes a closed part of the whole story with a visible outcome (for example, the search for products in an online shop). In general, a scenario should focus on one specific scope, only. In case one wants to describe the same scenario for different scopes, each combination of scenario and scope should be treated as independent elicitation activities (see Chapter 1).

The information typically gathered during preparation includes: information on typical users, typical location in which the system is used, and important events or situations that should be considered within the scenario.

Furthermore, the documentation format of the scenario should be selected (see typical artefacts for a list of formats).

Application

The application of the scenario technique is simple: start developing the scenario. The development can take place in a group setting or as individual work. Group work is advisable if the context of the scenario requires various competences (for example, a complicated business process).

During the development of a scenario, one should always create different alternatives. This allows for the exploration of alternative versions, increasing the chances of optimal end results. If the scenario is used as an intermediate result (see result processing), it is not necessary to consolidate the developed versions into one final version. Presenting alternative versions of a scenario to stakeholders fosters discussion and also allows their opinion on the alternatives to be incorporated into the development process.

[7] See, e.g., https://www.merriam-webster.com/dictionary/scenario
[8] See, e.g., https://www.merriam-webster.com/dictionary/storyboard

Result processing

Scenarios can either become part of a requirements specification or an intermediate result. If the former, the scenario should be documented in accordance with the standards of the project. If the latter, further activities are required to derive requirements from the scenario. This step is a subsequent technique and not part of the scenario technique.

Typical artifacts

Scenarios are typically described in a written form. The simplest form is a prose text:

John Doe waits at the bus station "Market Street". It's cold and rainy and he wants to go home quickly. Unfortunately, the bus appears to be late. John wants to check the arrival time of the bus. He grabs his smartphone, opens the public transport app and selects the function "departures from my station". The app uses the smartphone functionalities (e.g. GPS) to identify the "Market Street" station as John's current location and presents the current timetable including the 11 minutes delay.

A more structured approach is a scenarios description with dedicated steps:

1. John Doe waits at the bus station "Market Street". It's cold and rainy and he wants to go home quickly. Unfortunately, the bus appears to be late. John wants to check the arrival time of the bus.
2. The user grabs his smartphone and opens the public transport app.
3. The app presents the menu to the user and the user selects the function "departures from my station".
4. The app uses the smartphone functionality (e.g. GPS) to identify the "Market Street" station as the John's current location.
5. The app presents the current timetable including the 11 minutes delay.

The difference between the continuous scenario and the structured scenario is the breakdown into individual steps. Each step typically refers either to one interaction between the user and the system, or to an activity of the user in the context of the system (step 1 in the example). Scenarios are always specific concerning people (John Doe), quantities (11 Minutes), place (Market Street station) and relevant context (cold and rainy). This helps to bring the scenario to life: to be a real story in a real context. That is important to let stakeholders understand what the solution is about and helps them to talk about very specific issues and topics.

Note:

Do not confuse scenarios with use cases [Cockburn2001]. A use case specification is a technique to document the generic interaction between an actor and the system. Scenarios, in contrast, are more like test cases (in fact they can form the basis for test cases). They can be seen as one specific instance of moving through a use case.

Storyboards have already been mentioned in the introduction and are a visual representation form for scenarios. Storyboards are useful if the scenario contains a lot of action and if the context can be more easily visualized than described.

Opportunities

Scenarios are a good and lightweight technique for early elaboration and validation of ideas in terms of processes and activities. They can be used to discuss and explore alternative ways of realizing a process in a system. Because of their lightweight structure, they are easy to develop and can change rapidly. As a rule of thumb, every development project should have a common – and explicit - understanding of the scenarios the system under development should support. The scenario technique is therefore a good candidate for early elicitation.

Challenges

Creating good scenarios requires good authoring skills (e.g. structuring the story or using expressive words to describe the story). Without these skills, scenarios often become boring and good ideas can get lost because of a weak presentation. A skilled writer also avoids overly detailed scenarios in which too much unnecessary information is presented to the reader.

Beyond authoring skills, the application of storyboards further requires basic drawing ability to create recognizable images for each step. This basic level is not difficult to reach: a simple storyboard consists of stick figures and simple sketches. The more difficult challenge may be overcoming inhibitions when showing one's sketches to other people!

Variants

Scenarios are typically described in a positive sense, i.e. the scenario ends with a positive outcome and the user achieves what he wants to achieve. A good variant for scenarios is to focus on the negative outcome and to describe what may happen if the scenario does not end in positive way. For example, in the bus station scenario above, the GPS might not work. What will the app present to the user? This way of looking at a scenario typically offers a lot of additional insights and new ideas for the system under development.

A third variant of scenarios are misuse cases [SiOp2005]. A misuse case focuses on an intentional misuse of system functionalities to do harm to the user or to other stakeholders. They are especially useful for eliciting requirements related to system security.

Instead of drawing storyboards, the team could act scenes and take photos which then might be deliberately transformed using filters in a painting program.

3.3 Thinking tools

The previous sections explain a number of common techniques that have proven successful in the elicitation of requirements. They describe ways to gather information and to produce artefacts for the documentation and communication of requirements.

This section focuses on certain supporting techniques. They are not used on their own, but in conjunction with the other elicitation techniques. We call them *thinking tools*, because they intend to stimulate a way of thinking, or to create a mindset that contributes to the success of the elicitation itself.

3.3.1 Thinking in abstraction levels

Abstraction levels are a powerful thinking tool in requirements elicitation [GoWo2005], [Lauenroth2014]. They are often used as a kind of process model to structure the elicitation work, i.e., first elicit requirements on the highest level only and continue with lower levels later.

Understanding abstraction levels as a thinking tool requires an understanding of the term «abstraction». According to the Cambridge Dictionary[9], abstraction is "*the situation in which a subject is very general and not based on real situations*".

[9] https://dictionary.cambridge.org/dictionary/english/abstraction, validated 2.7.2019

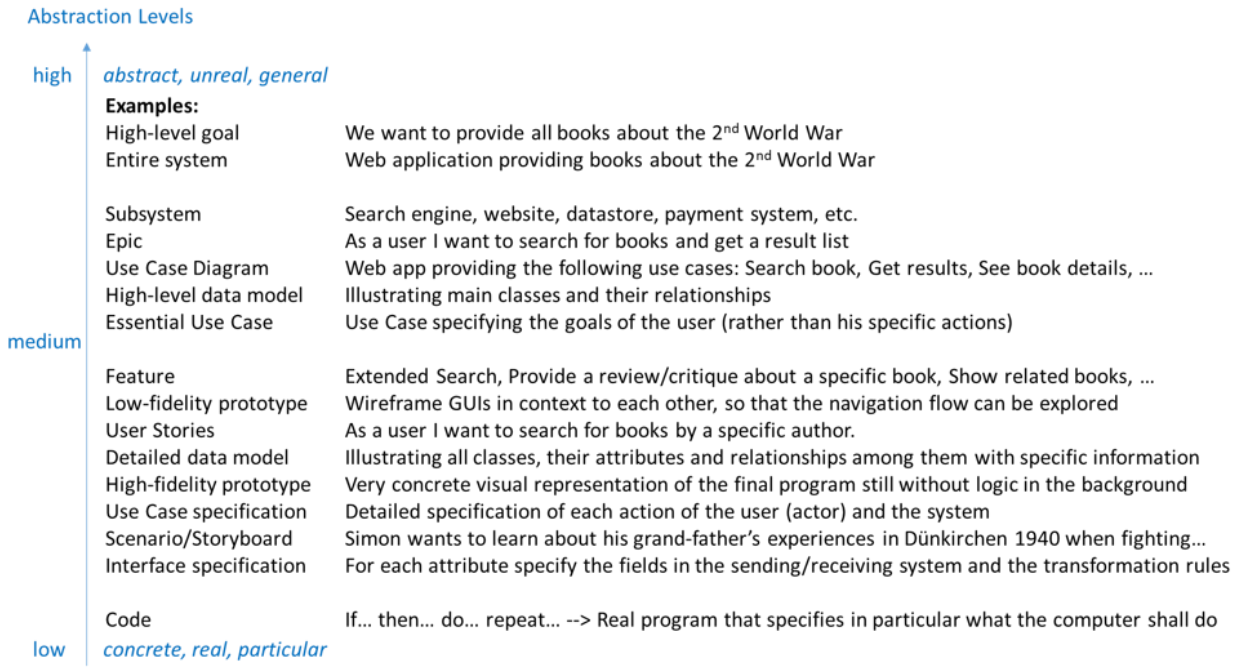

Figure 21: Representation of the concept of abstraction levels, illustrated with enlightening examples

In computer science, abstraction is achieved by information hiding [CoSh2007]. Programming languages and APIs are good examples for this. Each instruction in a programming language and each method in an API provide a specific functionality which is hidden behind the name of the instruction or method. In case somebody wants to know the details, it is possible to uncover them by reading the code of the function or method. The goal of abstraction is to reduce complexity by hiding detailed information behind a single, simpler construct. In the following, we will describe two examples for the application of abstraction layers in Requirements Engineering.

Three essential abstraction layers in Requirements Engineering

In Requirements Engineering, we have one natural intellectual border that can be used to reduce complexity: The border between system and context (system boundary, see [IREB2017]). The context deals with everything that is visible and can be experienced by the user or another system. This includes the user interface and technical interfaces. When we discuss the requirements of a system on the abstraction level of the system context, it is not necessary to talk about technical details that realize these requirements. Requirements Engineers, for instance, do this with a use case diagram or a user story map, visualizing the functionality provided by the system to the actors in the context while omitting all the implementation details behind.

The system itself can be further subdivided into a logical and a technical system. The logical system is an idealized description and includes logical data structures, functionalities and behavioral descriptions of certain elements at the system context level (e.g. the specification of the use cases mentioned above). The technical system refers to the realization in terms of hardware, technical data structures and software components (e.g., databases and frameworks). As an example, take the operating system of a computer: this is the API to the technical details of the hardware, such as how specific memory elements are addressed or how the processor executes individual operations, concurrently or otherwise.

The separation between system context, logical system, and technical system can be used to:

- Structure requirements specifications to make complicated content more accessible. For example, an interaction between a user and a system is described on the context level, whereas detailed data structure and functional requirements are specified on the system level in a later part of the specification. The reader of a specification may then decide if he/she wants to read the details (e.g., of a data model) or to skip the details in favor of an overall understanding.
- Structure the development process of a specification.
  For example, one extreme would be to follow a breadth-first approach by first understanding as much as possible on the context level before detailing the system level. An alternative extreme approach would be a depth-first approach which elaborates an aspect on the context level with all necessary details on the system level before continuing with the next element on the context level. For practical purposes, a mixture between both approaches is advisable. For example, success-critical aspects of a system should be elaborated with a depth-first approach whereas simple or properly understood aspects can be elaborated breadth-first because the details are clear anyway and the risk of omitting important details is low.

3.3.2 Thinking in terms of problems and goals

The development of a system is often based upon a certain problem as experienced by a client. On other occasions, the development is triggered by the wish of a client to reach a certain goal. But in most cases, clients are not very clear, or certain, nor open about their true problems and goals. Sometimes a client proposes a solution without even being able to explain what problem it would solve or what goal it should realize!

Therefore it is essential that the Requirements Engineer does not take for granted that a single problem or goal is the starting point for the development. A thorough analysis of the situation at hand is necessary to reveal the whole landscape of interrelated problems and goals before a viable solution can be designed (see [LoSL2017]). First, we need clarity about the definitions.

A *problem* is an aspect in the stakeholder's context that is currently experienced negatively. An anticipated negative experience in the stakeholder's context in the future is called a *risk* (i.e. a potential future problem). Often, a certain state in the context is perceived as a problem because it *inhibits* the stakeholder from doing something desirable or from achieving a goal.

A *goal* is an anticipated positive aspect in the stakeholder's context in the future. Often, a certain future state in the context is perceived as a goal if it will *enable the* stakeholder to do desirable things.

It is important to realize that problems and goals are *mental constructs*: they do not exist in the real world, but only in the minds of stakeholders. Therefore, they can only be found by elicitation. As a consequence, a certain (future) context state can even be a goal for one stakeholder, while it constitutes a potential problem for another.

Problems and goals are always connected to each other. A problem entails the goal of changing the negative state in a positive direction. A goal is only recognized as such if something in the present state prevents it from happening already.

Problems and goals are also connected to each other by another mental construct: the *solution*. A solution is the roadmap for an intervention in the context of the stakeholder: it describes a way in which the actual negative state in the present can be changed into a desired state in the future. Solutions are developed through a creative design process starting from the elicited problems and goals.

Explicit problems and goals may be occurrences of implicit higher or lower level problems or goals. They never come alone: they are part of a large family of parents and children. The siblings in this family must be discovered to get a complete picture.

Parents of a certain problem can be found by looking for causes. *What* causes this problem?
Parents of a goal can be revealed by analyzing the behavior that is enabled when the goal is reached. *Why* does the stakeholder have this goal?
The children in the family (the lower level problems and goals) can be found through solutions. Every action of a feasible solution sets a new goal (and problem) at a lower level for someone who is responsible for implementing it and has the challenge of how to effectively do so.

The following class model shows the relation between these concepts. A problem inhibits a goal and is caused by higher level problems. A goal enables higher level goals. A problem may be solved by a solution that may achieve the pertaining goal. A solution defines certain actions to perform. Each action sets one or more lower level goals on its implementation.

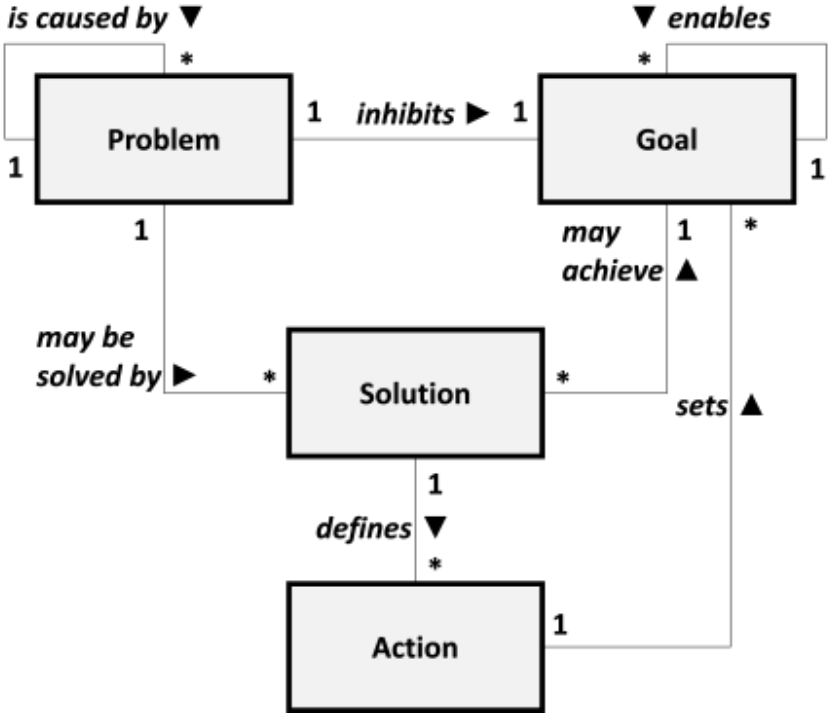

Figure 22: The relation between Problems, Goals and Solutions

Thinking in terms of problems and goals is not a itself technique. It is a mindset to encourage the Requirements Engineer to dig deeper into the context of the stakeholders before jumping to a seemingly obvious, but maybe inadequate, solution. This mindset is relevant for all kinds of elicitation techniques, whenever a problem or a goal is encountered.

For a problem, try to find out:

- Is it a problem in the present context (of which stakeholder) or is it an expectation of a future problem (then think of ways to prevent this future from becoming reality)?
- What situations in the present or past caused this problem (this may give you ideas about possible solutions)?
- What future actions are inhibited by the problem (the connected goal)? What would which stakeholder (be able to) do if the problem did not exist?

For a goal, think of:

- What situations in the present (or expectations about the future) inhibit the goal from being reached (the connected problem)? What will happen if no action is taken?
- What future actions will be possible when the goal is reached? Who will gain what benefit (the value)?

For solutions:

- Be careful when a client directly comes up with a certain solution. Make sure you really understand all connected problems and goals. Verify that the suggested solution solves the problem and realizes the goal.
- Consider the value: the balance between the expected benefits of a solution, the (total) costs for implementing it and the risk of failure.

Analyzing the whole landscape of connected problems and goals helps you find a solution that brings the highest overall value. A solution for a problem or goal mentioned by a stakeholder can bring value to that person but may be a risk for the company as a whole. As the goal of one stakeholder may cause a problem for another, thinking in terms of problems and goals may also help you in identifying and solving requirements conflicts.

Stakeholders often tell only part of the whole story when discussing problems and goals. As problems and goals are mental constructs, present only in the minds of certain stakeholders, they are by nature subjective which is hard to reveal. If you as a Requirements Engineer have difficulties in understanding the nature of a problem or the value of a goal, subjective components may be hidden underneath. *Why* questions may help to clarify them.

3.3.3 Avoidance of transformation effects

Due to varying levels of knowledge, different cultural and social backgrounds or professional experience, stakeholders may perceive the same information in different ways. The reality or a requested functionality of a system will be filtered by the personal perception of an individual, transformed into personal knowledge and later expressed in a more or less well-formed statement. The statement represents the knowledge and ideally a high amount of relevant information about the reality. One can distinguish two different types of transformation [BaGr2005] [Rupp et al.2014]:

Perceptional transformations occur since every person perceives reality in a different way and creates an individual image of it.

Representational transformations occur due to a conversion that happens as soon as a person expresses their knowledge in natural language.

These transformation processes can lead to a loss and distortion of information, which the Requirements Engineer has to reveal in order to document a complete set of high-quality requirements. Figure 23 describes the transformation processes.

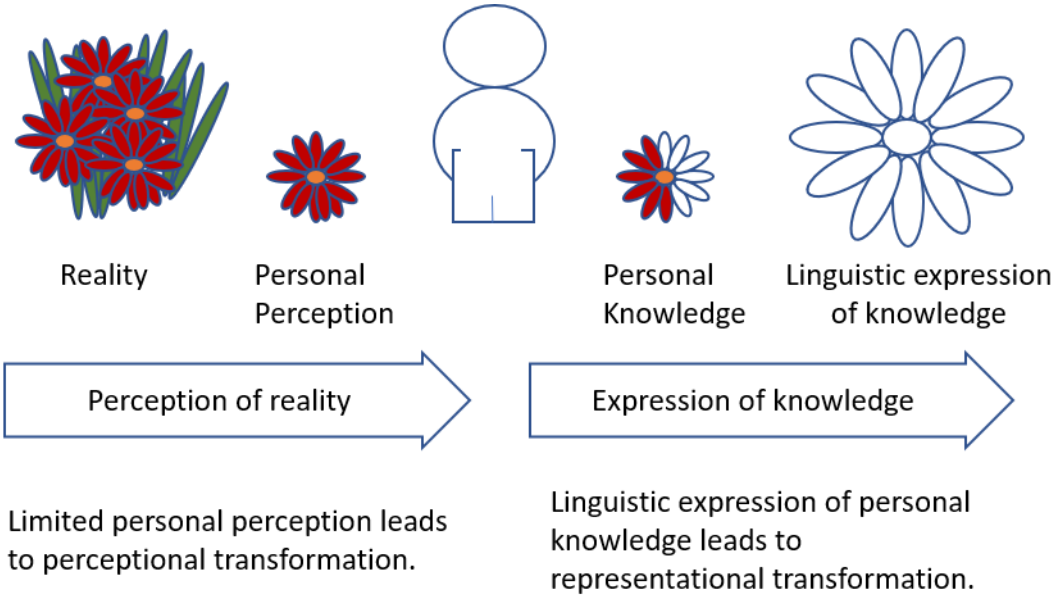

Figure 23: Transformation processes

In order to find out what information has got lost or distorted, the Requirements Engineer has to know the different categories of transformation processes.

Deletion is a process by which we selectively pay attention to certain dimensions of our experience and exclude others. Deletion reduces the world to proportions which we feel capable of handling [BaGr2005]. Deletion is an indicator that the Requirements Engineer has to elicit missing information.

Example:

A software developer might primarily consider the logical information (customer data) of an interface and may not be aware of all the physical information (material of the plug) required.

Generalization is the process by which elements of a person's model become detached from their original experience and come to represent the entire category of which the experience is an example [BaGr2005]. Generalization is an indicator that the Requirements Engineer has to elicit the missing information in order to decide what information can be generalized and what is in fact only valid in special circumstances.

Example:

For one stakeholder an address of a customer always contains the street name and number, but in another country street numbers are not customary.

Distortion allows us to fit an event or occurrence into a framework of pre-existing knowledge. It changes our interpretation of events to fit our existing understanding [McGrBa2017]. People substitute parts of the reality with their personal belief system. Distortion is an indicator that the Requirements Engineer has to elicit the missing facts in order to decide which of the statements are representing the reality.

Example:

A stakeholder may be convinced that two systems operate using the same database, as he/she observes that updated data is instantly available in both systems. In fact, there are two separate databases that are synchronized very quickly.

Each of the transformation categories mentioned above are part of our communication and are used every day. In a familiar context, or working with known stakeholders, transformation processes may be less likely, but in general a new system or new functionality will involve new knowledge.

In order to avoid effects resulting purely from communication the Requirements Engineer should apply some basic measures in written and verbal communications [Rupp et al.2014]:

- Always express statements in complete sentences
- Always use the active voice in your sentences
- Use terms defined in the glossary
- Use consistent terminology and avoid synonyms or homonyms
- Express processes via main verbs

For more information about communication in Requirements Engineering, see Section 5.2.

Hint 3.3.1:

Include the finding of suitable definitions in your elicitation activities. In particular, agreed process verbs can help to develop more unified communication. Term models (e.g. with UML Class Diagrams) can help in addressing the correct dataset in a requirement.

Hint 3.3.2:

Using requirements templates will reduce transformation defects, as a template requires basic information to be completed in the requirement.

Requirements Engineers can make use of the SOPHIST Set of *Re*gulations, as described in [Rupp et al.2014], which help in analyzing statements (or requirements) to reveal defects based on the transformation categories (deletion, generalization, distortion).

One rule (out of eighteen) covers one or more transformational effects. The Requirements Engineer will look for certain signal words (step 1: Identify) and will derive questions for the stakeholder from the signal word (step 2: Analyze). After the stakeholder has answered the questions the Requirements Engineer will correct or complete the requirement (step 3: Resolve). The following rules are instances of transformational effects mentioned in the IREB CPRE Foundation Level syllabus [IREB2017].

Hint 3.3.3:

It may be useful to include a certain rule in your personal development plan and try to implement it in communication, documentation and reviews of requirements. After four weeks you can try to apply the next rule.

Resolve nominalizations

Nominalizations can blur the process (or process steps) addressed in the requirement, so it is unclear how the process shall be performed. The information is distorted (Distortion).

Example statement: “The library system shall offer archiving.”

Step 1: Identify

Archiving is a nominalization (also the signal word) of the verb "to archive". To archive data is functionality with defined steps. The steps are not yet clear, neither are the process details (e.g. when and how the process starts).

Step 2: Analyze

The question to be answered by the stakeholder might be: "What are the process details of 'archiving'?"

Step 3: Resolve

In this example the stakeholder (the librarian) will perform the process steps "choose customer data" and "archive the chosen customer data" manually.

"The library system shall provide the librarian with the ability to **choose customer data** for archiving."

"If a customer currently has no borrowed items, the library system shall provide the librarian with the ability to **archive the chosen customer data**."

In this example the nominalization was cleared so the two process steps were documented in two separate but interdependent requirements.

Nouns without reference index

If a stakeholder statement contains nouns without a reference index it is not clear which of the objects or actors are addressed. The formulation is too general to be implemented (Generalization).

Example statement: "The system shall display the data to the user."

Step 1: Identify

In this example it is unclear which **data** shall be displayed with which **user**.

Step 2: Analyze

The question to be answered by the stakeholder might be:

- "Who is the user that shall read the data?"
  Answer: "The Librarian."
- "What data shall be displayed to the Librarian?"
  Answer: "all statistically calculated data of library items."

Step 3: Resolve

"The library system shall display **all statistically calculated data of library items** to **the librarian**".

Hint 3.3.4:

In order to elicit the reference index, you can review the roles defined in the stakeholder list or the role definition of the system. In order to find the correct dataset, you can review the data model of the system.

Hint 3.3.5:

Sometimes it can be useful to define a more general term for a superset of data. For example, registration data and payment details can be generalized to customer data, which is then defined in the glossary.

Universal quantifiers

If a stakeholder statement contains universal quantifiers, the quantity of objects may be either too general or too specific, and has to be adapted.

Example statement: "The library system shall provide each customer with the ability to change all customer data."

Step 1: Identify

Depending on the definition of customer data, the requirement could be formulated correctly. But in this case the data set and the reference to the customer are questioned.

Step 2: Analyze

The question to be answered by the stakeholder might be:

- So, customers can change every piece of data ever saved in the library system?
  Answer: "No, of course not. Only his password and his profile registration data."
- "Meaning every single customer can change customer registration data of all customers?"
  Answer: "No, of course not! A customer can only change **her own registration data**."

Step 3: Resolve

"The library system shall provide **each** customer with the ability to **change his or her customer registration data.**"

In the end **each** customer was identified as the correct quantifier. The amount of customer data was limited to **customer registration data** and the reference that a customer could change besides **his or her own customer registration data** was corrected.

Incomplete conditional structures

If a requirement contains conditions, they can appear to have more than one aspect. In order to complete the requirement, all relevant conditions should be explored. Otherwise some aspects remain deleted (Deletion).

Example statement: "If an item is not damaged and not reserved, the library system shall provide the librarian with the ability to continue the loan process."

Step 1: Identify

The signal word "if" indicates a condition. The signal word "and" indicates the condition has more than one part.

Step 2: Analyze

What happens if the item is damaged and/or reserved?

How does the system behave?

Step 3: Resolve

If the item is **not damaged and reserved**, the library system shall display an error message to the librarian.

If the item is **damaged and not reserved**, the library system shall provide the librarian with the ability to stop the loan process.

If the item is damaged and reserved, …

In the end all combinations of conditions have to be revealed and the Requirements Engineer has to elicit the behavior of the system for each path.

Analyzing requirements in detail will minimize the risk that the understanding of the reality might be wrong, but the Requirements Engineer must also consider the cost of the elicitation activity. Depending on the abstraction level (see 3.3.1), the progress of the project and the characteristics of the stakeholders, it might be acceptable to tolerate some inaccuracy.

Hint 3.3.6:

If communication seems to be difficult or mechanisms appear to be complicated, ask a least two different stakeholders to provide information about the given subject.

3.3.4 Thinking in terms of models

The IREB CPRE Foundation Level syllabus [IREB2017] introduces several types of models (e.g. data flow diagrams, activity diagrams) for documenting requirements. Models allow focusing on a specific perspective of a system: data, function, behavior. This focus is not only applicable to the documentation of requirements, it can also serve as a thinking tool during requirements elicitation. The Requirements Engineer can select a particular model and concentrate on the perspective provided by that model; the model can be either an explicit or an implicit thinking tool.

When a model is used as an *explicit thinking tool*, the Requirements Engineer develops the model together with the stakeholders. The Requirements Engineer should keep in mind that models are only useful in such situations if the modelling language is understood well by all involved stakeholders.

For example, a Requirements Engineer wants to elaborate a specific business process to be supported by a system. This elaboration activity could take place in a workshop using activity diagrams. The activity diagram representing the business process is developed together with the stakeholders, e.g. by drawing the activity diagram on a whiteboard or large flip charts. In such a situation, the activity diagram notation serves as a toolkit for what can be expressed and documented during the workshop. The Requirements Engineer must pay attention to the content that is developed during the workshop, since stakeholders are typically not very familiar with modelling. If the stakeholders do not apply the selected notation properly, the Requirements Engineer must provide support to the stakeholders to create a proper model. Such mistakes often occur due to stakeholders wanting to express an important piece of information that does not fit into the model.

Hint 3.3.7:

Information that does not fit into a selected modeling notation should not be disregarded. For example, if a group of stakeholders is developing an activity diagram, requirements related to data structures often come up. These requirements cannot be documented properly in an activity diagram. In order not to lose these requirements they should be saved, for example in the workshop protocol, to allow for later analysis.

Models can also be used as an *implicit thinking tool*. In this situation, the Requirements Engineer uses a particular modelling language to structure his/her own thoughts during requirements elicitation. The model does not become an explicit part of the elicitation activity and is not discussed with the involved stakeholders.

Instead, the Requirements Engineer uses the information obtained during the elicitation activity to develop the model and uses the created model as a reflection point for his/her own thought and as a cue to ask further questions or to search for additional information.

For example, a Requirements Engineer eliciting requirements for an online system for selling accident insurance wants to understand the data that is needed to apply. He/she decides to interview accident insurance experts and to analyze existing paper-based application forms.

He/she can start with the application forms and derive a simple data model from the content of the application form. The result of this analysis is an initial data model which contains entities that are not fully clear to the Requirements Engineer, that is missing relationships between entities and which has an incomplete set of attributes. The Requirements Engineer now can use the incomplete model to prepare for the interviews with the insurance experts. Instead of showing the model during the interview, the Requirements Engineer uses the model as guideline for the interview. The answers given by the experts can now be mapped onto existing, known elements in the model, allowing the Requirements Engineer to more easily identify gaps and ask additional questions to clarify these areas.

Hint 3.3.8:

Models that are created as a thinking tool should not be confused with documented requirements. The thinking tool is an intermediate result and will be used to develop detailed, documented requirements in a subsequent activity. It is therefore advisable to throw thinking tool models away (or put them into an archive) as soon as the insights from the model have made their way into a requirements document. Otherwise, there is a risk that the thinking tool model will cause confusion in the project as such models are typically not maintained and are soon outdated or become inconsistent with requirements documents.

3.3.5 Mind mapping

Mind mapping is an activity in which a concept is visualized in a so-called mind map. In other words, a mind map is a graphical thinking tool [BuBu2005]. By putting a main topic in the center and spreading out the ideas in branches, thoughts and ideas can be sorted and restructured. Text and images should both be used as well as color. “Boring” representations (straight lines, only one color) should be avoided to make the representation more “stimulating” for the brain.

The idea is based on studies of how the human brain works. Instead of a linear or lateral representation, as in books or lists, the brain organizes knowledge in a multi-dimensional way, also called “radiant”. A mind map is an expression of radiant thinking that supports the natural thinking process [BuBu2005]. The essential characteristics of a mind map are:

1. The subject of attention is crystallized in a **central image** of each mind map.
2. The main themes of the subject “radiate” from the central image as **branches**.
3. Branches comprise **key images or key words** and are refined by more branches representing **subthemes**.

During the mind mapping, a structured mind map with **hierarchies, visualizations** and **associations** between branches is created, as in Figure 24.

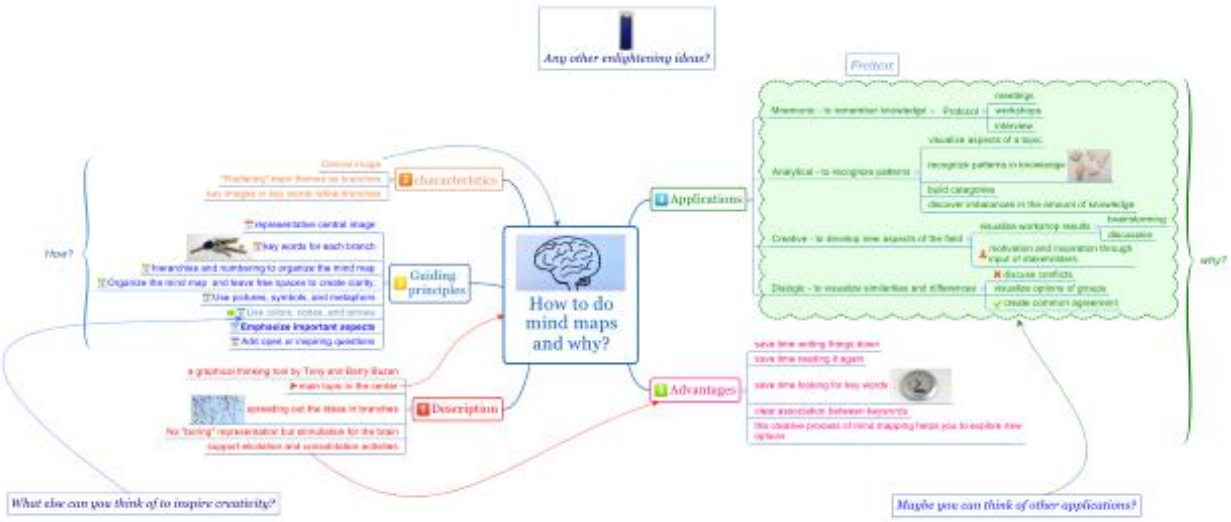

Figure 24: Example of a mind map with the content of this subchapter

There are several applications of a mind map that can also support elicitation and conflict resolution activities.

Mnemonic - to remember knowledge

- A mind map can be used as a visualization of a protocol in a meeting or a workshop. It can even be created simultaneously during the interview, as it only covers the most important keywords. It also creates an overview of the main topic that has to be covered in a meeting.

Analytical - to recognize patterns

- It can be very useful to visualize different aspects of a topic and recognize patterns in knowledge. This way the Requirements Engineer can build categories or discover imbalances in the amount of knowledge.

Creative - to develop new aspects of the field

- The mind map can also be used in workshops to visualize the results of a brainstorming or a discussion. The mind map is a fast and motivating way to involve the stakeholders and encourages inspiration through the input of others.

Dialogic - to visualize similarities and differences between concepts and minds

- If stakeholders have different opinions it can be useful to visualize the main points of different individuals and discuss them in a group. Afterwards the stakeholders can create a common mind map to document the agreements.

In order to benefit the most from the mind map as a thinking tool, the author should follow some guidelines.

Guiding principles of mind mapping:

- Use a representative central image for the topic in the center of each map;
- Use key words for each branch or topic;
- Use hierarchies and numbering to organize the mind map;
- Organize the mind map to keep the overview and leave free spaces to create clarity;
- Use pictures, symbols, and metaphors to address all human senses;

- Use colors, codes, and arrows to visualize associations between subtopics and branches;
- Emphasize important aspects through variation in font, line styles, color and size of the visualization;
- Use multiple dimensions in the structure of your map;
- Use colors and clouds to frame branches that belong together;
- Add open or inspiring questions to your mind map.

For more guidance see [BuBu2005].

The Requirements Engineer can use all kinds of tools (e.g. canvases or boards) to create mind maps. A variation of mind mapping is to use a computer-based tool to document results digitally and present them on a projector. This makes reorganizing the mind map much easier. Indeed, with such tools simultaneous editing of the same mind map from anywhere in the world may be possible, increasing opportunities for cooperation.

The branches of the mind map often contain important terms for the glossary or term model, and associations between branches can be documented as "Dependencies" between objects.

3.4 Describing elicitation techniques by attributes

Requirements Engineers should carefully select which elicitation techniques to use based on the specific context and needs of the current situation. Many researchers propose models and frameworks to map elicitation techniques to the given circumstances. For example, Carrizo et al. provide extensive research on the systematic selection of elicitation techniques [CRCN2014], while Tiwari and Singh propose a methodology for the selection of requirement elicitation techniques and nicely summarize related research on this topic [TiSi2017].

In the approaches mentioned above, the authors often focus on the variability of the problem at hand (project, users, context, etc.), but provide a direct mapping to suitable elicitation techniques.

For the purpose of learning the requirements elicitation techniques without having to remember by heart all the pros and cons of each one in detail (i.e. not as suggested by [YoAs2015]), we introduce *identifying attributes* that help to classify elicitation techniques. With this additional layer between the problem space (how to elicit requirements in a given circumstance) and the solution space (hundreds of available elicitation techniques), the learner can focus on the main problem solving concepts in requirements elicitation: that is, which identifying attributes an elicitation technique must have to achieve the elicitation objective (see Section 0).

Table 3.4.3-1 below provides a list of identifying attributes. We define an attribute as *identifying* if it is an essential property of an elicitation technique.

The characteristics of each elicitation technique can be described by a combination of these attributes. For example, the technique of *interviewing* is characterized by the attributes *conversational* and *questioning*. An interview might also be *observational*, in the case that the Requirements Engineer conducts the interview at the location of the end user. However, *observational* is not a core (i.e. identifying) attribute of interviews, as they could also be conducted by phone or at other locations where observation is not possible.

Classifying a long list of available techniques by relevant attributes can help in selecting the right techniques in a given situation. Table 3.4.3-2 provides such a classification for a subset of techniques.

"There are good practices in context, but there are no best practices" [KaBP2002]: every situation requires a particular combination of elicitation techniques identified by the relevant attributes for that situation.

Or, to put it in other words, given the elicitation objective, project situation, stakeholders, etc., the relevant attributes should be determined and the elicitation technique(s) matching exactly these identifying attributes selected. An example of such a mapping between context and attributes is provided in Table 3.4.3-3.

Table 3.4.3-1: Attributes for classifying elicitation techniques

Attribute	Short description	Aiming at the following goals	Suitable in the following situations
Conversational	A dialogue between Requirements Engineer and stakeholder(s)	To understand the system context; to elicit goals and obtain an overview of satisfiers (Kano)	When (relevant) stakeholders are available for oral information exchange
Questioning	Asking stakeholders (at least partly) prepared questions to learn about facts or about their opinion	To elicit goals and satisfiers; to verify dissatisfiers; to obtain stakeholder's opinion or additional information on previously elicited requirements; to elicit detailed information; to clarify specific requirements	If relevant questions can be formulated upfront; if some form of communication with stakeholders is possible; if complicated subject matter is concerned
Observational	Observing stakeholders' behaviors in a live situation, usually operating an existing system or performing specific tasks	To gather information about the stakeholder's actual behavior; to elicit dissatisfiers; to analyze usability requirements	If stakeholders cannot be addressed directly or if they are unable to state their needs and actions (detailed enough); when in doubt on congruence between actual and stated situation; to improve understanding the users' needs; to improve understanding of the project (e.g. in preparation for other elicitation techniques)

Attribute	Short description	Aiming at the following goals	Suitable in the following situations
Provoking (dis-) agreement	Demonstrating relevant aspects of a solution in order to get affirmative or contradicting feedback from stakeholders	To make requirements tangible for stakeholders; to evaluate previously elicited requirements; to get feedback on variants of a solution	If stakeholders have trouble imagining things; if the Requirements Engineer can explain or show aspects of the proposed solution to the stakeholders (or even let them use it); if stakeholders have trouble explaining what they need
Artefact-based	Gathering and analyzing existing artefacts (e.g., documents, models, products or systems in use)	To derive requirements from existing artefacts; to elicit (dis-)satisfiers, especially constraints	When relevant artefacts are available and accessible; to improve understanding of the project and of the domain (e.g. in preparation for other elicitation techniques); if stakeholders are not directly available
Creativity-stimulating	Foster creativity and innovation	To elicit delighters; to come up with novel approaches	If innovation is needed; if a predetermined direction is absent; when other approaches fail
Experiencing	Experiencing the environment and problem space where the system to be developed will be used	To derive requirements from the real-life circumstances; to understand the problem to be solved from users in their work context; to gain empathy	If users and usability are key aspects of the project; when it is possible to access the environment where usage actually takes place

Table 3.4.3-2: Subset of elicitation techniques described by their identifying attributes

Techniques	Attributes						
	Conver-sational	Questioning	Obser-vational	Provoking (dis-) agreement	Artefact-based	Creativity-stimulating	Experiencing
Interview	I	I					
Questionnaire		I					
Requirements Workshop (e.g. Focus Group)	I						
Field Observation			I				I
Apprenticing	I		I		I		I
Contextual Inquiry	I	I	I		I		I
Creativity Techniques (Brainstorming, ...)						I	
System Archaeology					I		
Perspective-based reading					I		
Requirements Re-use					I		
Prototyping				I			
Scenarios				I			
Storyboards				I			

Techniques	Attributes						
	Conver-sational	Questioning	Obser-vational	Provoking (dis-) agreement	Artefact-based	Creativity-stimulating	Experiencing
User Walkthrough	I			I			
Usability Testing			I	I			
Requirements guessing				I		I	
User Story Elaboration (Card, Conversation, Confirmation)	I						
Diary study					I		I
Card Sorting			I				
...							

Legend:

I	**Identifying attribute**, i.e. it is a key property of the technique – without that attribute it is a different technique.
<empty>	The attribute is not a core property (i.e., **no identifying attribute**), although it may be possible for the technique to fit this attribute under certain circumstances.

Table 3.4.3-3: Elicitation objectives, constraints and project situations that determine identifying attributes which suitable elicitation techniques should contain:

Research question / Constraint / Project Situation	Identifying attribute(s) of techniques you might apply
If you need to elicit stakeholders' implicit knowledge	Observational
If it is not possible to interrupt users working on a task	Observational
If relevant aspects of a preceding system are not clear or play an important role	Observational and/or Conversational
If you already have (parts of) an existing system, preceding systems or similar systems or if you already produced possible solutions or mockups, e.g. by prototyping	Provoking (dis-) agreement
If the environment in which the future system will be used is an important source for requirements (noise, time pressure, physical conditions etc.)	Experiencing
If you need to elicit important usability requirements	Experiencing
If your project is operating in uncharted territory	Provoking (dis-) agreement Creativity-stimulating
If no stakeholder is able to specify requirements on a specific level of detail	Provoking (dis-) agreement
If you develop a complex system that shall be used by casual users	Provoking (dis-) agreement
If you identified non-human requirements sources	Artefact-based
If you want to find out where process flaws are	Artefact-based and Observational and Experiencing
If you want to find out how and with what artefacts your future users really work	Experiencing and Artefact-based
If you need material to prepare for elicitation activities with stakeholders	Artefact-based
If you need a catalyst for stakeholders to come up with requirements they otherwise wouldn't have remembered or been aware of	Artefact-based
If you develop an innovative new system that needs to provide new features or new ways of interaction (i.e. If you are looking for delighters according to the Kano-Model)	Creativity-stimulating

Research question / Constraint / Project Situation	Identifying attribute(s) of techniques you might apply
If the solution to your problem is not obvious and you and your team need to come up with a new approach	Creativity-stimulating
If the Requirements Engineer needs to specify the solution rather than just listening to stakeholders and writing down the requirements he gets told	Experiencing
If the REs shall continue to maintain the solution after completing the initial implementation (i.e. the increased and founded know-how gained by the RE during his elicitation activities is in itself a valuable asset to the sponsor)	Experiencing
If in specific project setups relevant stakeholders are rarely available and it is therefore easier and/or faster to study the domain under investigation and for Requirements Engineer to come up with requirements themselves (and just validating requirements with the stakeholders)	Experiencing
If you need to clarify requirements regarding the current situation, needs and possible solutions	Conversational
If you want to make sure that you speak the same language with stakeholders, especially in specialized domains	Conversational
If the degree of user interaction is high	Experiencing Provoking (dis-) agreement
If the degree of technical integration is high	Artefact-based
If the level of innovation (=> Kano) is high	Creativity-stimulating Provoking (dis-) agreement

4. Conflict resolution

During elicitation, the Requirements Engineer gathers a broad collection of requirements, often from different sources (see Chapter 2), with different techniques and at different levels of abstraction and detail (see Chapter 3). Elicitation techniques by themselves do not ensure that this collection as a whole meets all quality criteria for requirements (see [IREB2017]). If quality criteria are not met, additional elicitation activities may be required in order to improve quality.

Additional activities are necessary to turn this collection into a single, consistent set of requirements that captures the essence of the system. Often it is found that some requirements are conflicting: inconsistent, incompatible, contradictory. Usually, this is caused by disagreement between certain stakeholders. As a consequence, "agreement" is a highly important quality criterion that should always be checked: all stakeholders have to understand and agree on all requirements that are relevant to them. If some stakeholders do not agree, this situation should be recognized as a requirements conflict to be resolved accordingly.

By definition, the goal of Requirements Engineering is to achieve "… a consensus among the stakeholders about [these] requirements" (see [IREB2017]). A major task in this respect is the handling of requirements conflicts.

Conflict resolution in the broad sense consists of four tasks:

- Conflict identification
- Conflict analysis
- Conflict resolution
- Documentation of conflict resolution

Conflict identification and analysis is an ongoing activity in Requirements Engineering and is a prerequisite for resolving any conflict. Once a requirements conflict has been identified, the Requirements Engineer should initiate conflict resolution activities to select a proper resolution technique and to document its outcome.

In our everyday life we are often involved in conflicts. As they are typically not pleasant to deal with, a common strategy is simply to escape from them. As a consequence people do not talk to each other, look for different working areas or even change the project or job so they are not involved with the conflict anymore.

Dealing with requirements conflicts can be stressful and time-consuming, especially if it contains personal issues. But it is essential to consider the following aspects:

1. Solving personal issues is not part of the job description and has to be escalated through different management activities. See Section 4.2 – Conflict analysis.
2. Escaping requirements conflicts is not an option, as unresolved requirements conflicts result in low quality requirement documents and frustrated stakeholders.

A constant awareness of conflicts and a regular application of reviews [IREB2017] will help in discovering conflict indicators and collect data for conflict resolution.

4.1 Conflict identification

Conflicts in general are a subject of social sciences and typically referred to as "social conflict" to indicate that a conflict arises between people. A social conflict can be defined as follows:
[Glasl1999] F. Glasl: Confronting Conflict - A first-aid kit for handling conflict. Hawthorn Press, Gloucestershire, 1999.

[Glasl2004][10] defines a social conflict as "an interaction between agents (individuals, groups, organizations etc.), where at least one agent perceives incompatibilities between her thinking/ideas/perceptions and/or feelings and/or will and that of the other agent (or agents), and feels restricted by the other's action."
A requirements conflict can be interpreted as a special type of social conflict and is defined as follows: "A conflict in Requirements Engineering (requirements conflict) is an incompatibility of requirements, based on a contradictory perception of two or more stakeholders." [Rupp et al.2014] There are several indicators by which conflicts can be detected. Indicators can be observed in communication and documentation.

Commonly encountered indicators in communication are:

- Denial: A stakeholder always finds fault with all propositions and always finds a tiny problem which he uses to argue against every suggestion.
- Indifference: A stakeholder does not (want to) contribute in a discussion or approves without critical questioning.
- Pedantry: A stakeholder always finds fault with all propositions and always finds a tiny problem which he uses to argue against every suggestion.
- Questions of detail: A stakeholder uses his position or his knowledge to question all statements very critically. It may look like the stakeholder wants to ensure that important requirements are not forgotten, but it in fact slows down the elicitation process.
- Incorrect interpretation: A stakeholder misinterprets facts on purpose in order to confuse or slow down the elicitation process.
- Concealment: A stakeholder consciously or subconsciously hides information and only shares information on demand.
- Delegation: A stakeholder commits himself to statements only loosely with the demand that others should state them in greater detail.

Commonly encountered indicators in documentation are (with examples):

- Contradictory statements by stakeholders: During workshops, stakeholders agree on a requirement that is not consistent with a requirement derived from an interview protocol of a previous elicitation activity.
- Conflicting results from analysis of documents or systems: The interface specification of a system contains a temporary address for a customer, but in the system, it is only possible to enter a second main address.
- Inconsistent requirements in detail: The system identifies duplicated customer data sets via name, day of birth and address, but there are some customers that have no address.
- Inconsistent usage of terms in specification: stakeholders use the terms customer, user and employee with different meanings, but do not apply the definitions from the glossary.

Most conflicts tend to be hidden and can only be detected by carefully monitoring these indicators. If one of the indicators occurs, this does not mean that a requirements conflict is present. However, the Requirements Engineer should continuously pay attention. Through most of the requirements elicitation activities, she/he is encouraging the stakeholders to state their positions clearly, thus revealing unexpected problems or existing conflicts.

[10] Citation translated from German based on [Glasl1999]

4.2 Conflict analysis

If there are absolutely no signs for requirements conflicts the Requirements Engineer should be suspicious and plan some reviews. As soon as a conflict is suspected, he/she can include data collection for a conflict in requirements elicitation activities.

Once a conflict has been identified, the Requirements Engineer has to clarify whether or not the identified conflict is a *requirements* conflict. This distinction is important since the resolution of a requirements conflict is the primary responsibility of the Requirements Engineer, whereas other conflicts have to be resolved by other participants (e.g. a project manager).

Analyzing the characteristics of a requirements conflict helps the Requirements Engineer to understand its nature. During conflict analysis there are various indicators for conflicts. It can be useful to collect indicators first and revise them later. With more information it is much easier to find an appropriate resolution of the conflict.

The following characteristics [Rupp et al.2014] of a conflict can help to understand its nature and to find an appropriate resolution.

4.2.1 The characteristics of a requirements conflict

The *type of conflict* determines its characteristics and the degree of personal involvement of the stakeholders. It is one of the most important indicators based on which conflict resolution techniques shall be excluded or applied (see Section 4.3 – Conflict resolution). In some cases, it might be hard to determine the type of conflict. If this is the case multiple types of conflict should be considered.

Subject matter is the problem behind the conflict. Finding out what the real issue behind the discussion is can be very difficult, depending on the type of conflict and its history. But it is also very valuable for a proper resolution. The Requirements Engineer should remain neutral with respect to the available options and can actively facilitate the analysis by mirroring the statements to the participants.

Affected requirements are the representative statements for the conflict. They can be used for analysis and to visualize details. Once a resolution for the conflict is found, the documentation of the relevant requirements should be easy.

Hint 4.2.1:

The Requirements Engineer can ask your stakeholders to formulate their concerns as requirements and let other conflict parties confirm or reject the statements. This will support precise and reflective communication.

Involved stakeholders can be the authors or others in some way responsible for the affected requirements. They are the sources of the information for analysis and may themselves be part of the conflict. Sometimes it can be helpful to involve more stakeholders to provide expertise and moderate or even solve the conflict between the parties by means of their authority.

Opinions are the statements of the stakeholders or a verbal summary of the concepts they have in mind. As a Requirements Engineer one can rephrase (or let them be rephrased by the stakeholders) in front of the other involved parties. Vague or unclear aspects can be explained to help the conflicting parties understand the underlying issue.

The *cause* is the reason why the stakeholders cannot keep on working independently. Once this is clarified, it may show the way to a proper negotiation technique or even the resolution.

The *history* of the conflict can help new parties to understand past approaches or arguments against affected requirements or options. It is likely that not all parties have the same level of knowledge, which can in fact be the main reason for the conflict.

Consequences are the costs associated with the relevant implemented requirements. These costs may also be unclear to some of the parties and should be estimated in order to contribute to the conflict resolution.

Resulting risks can be very important in order to decide when (i.e. in which phase of the analysis) to solve the conflict.

Project constraints are of personal, organizational, content-specific or domain-specific nature. They are related to the type of conflict and influence the choice of a suitable conflict resolution technique. In a particular project situation, for example, there might not be enough time (organizational project constraint) to solve the conflict with the technique agreement (see Section 4.3 – Conflict resolution).

4.2.2 The conflict types of Moore

The type of the conflict is important for deciding if a given conflict is a requirements conflict or not. Five different types of conflicts can be distinguished [Moore2014]:

- Interest conflict
- Data conflict
- Value conflict
- Structural conflict
- Relationship conflict

Most requirements conflicts can be categorized as either interest conflicts, data conflicts or value conflicts. Structural and relationship conflicts are usually not related to requirements and should therefore be resolved by other participants. However, most conflicts show characteristics of more than one type as different causes interact. Requirements Engineers should therefore pay attention to all kinds of conflict, even if a solution is not within their responsibility.

4.2.2.1 Interest conflict

An interest conflict is based on the different motivations of the conflict parties. Motivations can be formed by personal goals, goals related to a group or goals related to a role. As an interest conflict is not based on possession of information (as the data conflict), it is important to understand the concerns and the needs of the stakeholders to solve this type of conflict. In the case of personal interests, stakeholders often do not reveal their true motives but find artificial arguments.

Examples of interest conflicts:

- A stakeholder of the safety department may request higher standards on encryption which require more time, while the user of the system emphasizes the performance of the system for his/her daily work.
- A stakeholder wants his department to be responsible for the implementation of a function because of prestige for his employees, while the system architect argues for another component in order to improve the stability of the system architecture.
- A stakeholder requires a function for his work to be implemented in the next release, but the sponsor of the system believes other functions to be more important for the majority of users.

If an interest conflict is the reason for a discussion, one can observe that communication amongst conflict parties is focused on the "appeal" of the "four-sides" model of Schulz von Thun [Schulz von Thun 1981]. The conflict parties try to convince others to follow their arguments and understand the needs of the role or group.

Prevention and resolution strategies:

- Interest and consequences have to be revealed and considered in a different context or by a different instance that can be more objective.
- Out of the set of interests the facts relevant for the conflict may be extracted so a resolution based on facts can be supported.

4.2.2.2 Data conflict

A data conflict is based on a lack of, or an uneven distribution of knowledge, or on a different interpretation of the data available to the conflict parties.

Examples of data conflicts:

- Stakeholders may argue about the existence or the interpretation of a business rule that is described in a requirement.
- Stakeholders differ on the reason for a requirement.
- Stakeholders have a different understanding of the terms and their definitions contained in requirements.

If a data conflict is the reason behind a discussion one can observe that communication amongst the conflict parties is focused on the "factual information" of the "four-sides" model of Schulz von Thun [Schulz von Thun 1981]. Stakeholders exchange information and share important facts and figures.

Prevention and resolution strategies:

- Data has to be provided to the stakeholders by exchanging, or collection of, additional information.
- The information relevant to the specific data conflict has to be identified.
- Stakeholders should agree on a common data collection process with evaluation criteria.
- Stakeholders should agree on experts to provide relevant information.

4.2.2.3 Value conflict

A value conflict is based on different values and principles. It is related to the interest conflict, but is more individual and involves global or long-term perspectives. If a person changes her/his role, interests may change but values are more stable and rarely change in the short term.

Examples of value conflicts:

- Stakeholders may avoid products with a lot of plastic or that are not recyclable, while others prefer low cost products. The value of protecting the environment stands against the value of cheap production.
- Stakeholders may find graphical representation in software less important than command-based applications.
- Stakeholders may find price discrimination (= a provider sells the same product at different prices to different customers) unfair for the users of an e-business website.

If a value conflict is the reason for a discussion one can observe that communication amongst conflict parties is focused on the "self-revelation" of the "four-sides" model of Schulz von Thun [Schulz von Thun 1981]. The conflict parties emphasize why their arguments are important from their point of view and give a number of arguments that reveal their inner values and principles. They tend to insist on their arguments and seem to have experience in arguing for the respective issue.

Prevention and resolution strategies:

- Stakeholders shall allow the conflict parties to agree or disagree on arguments, without judgment but with tolerance.
- Conflict parties shall concentrate on their common ground where their values are aligned.
- Conflict parties shall concentrate on a common global goal and the bigger context rather than on their differences and the details.

4.2.2.4 Structural conflict

A structural conflict is based on inequality of power, competition over limited resources and structural dependencies which influence the conflict parties. The perceived imbalance causes problems in communication, eliciting requirements and decision making. Another reason for a structural conflict may be strict restrictions on resources or dependencies on work products to be delivered by other parties.

Examples of structural conflicts:

- Stakeholders may suppress requirements because they believe a conflict party with greater authority may argue against it.
- Stakeholders with greater influence in the organization may try to change the priority of requirements.
- Stakeholders wanting more transparency formulate a requirement for access to a specific application so that they cannot be surprised by another department always delivering information too late.

If a structural conflict is the reason behind a discussion one can observe that communication amongst the conflict parties is focused on the "relationship" of the "four-sides" model of Schulz von Thun [Schulz von Thun 1981]. Conflict parties may use the discussion on requirements to either change or preserve the status quo. Depending on the point of view, the conflict parties emphasize their disagreement with or desire to change (for power-holding party: preserve) the current structure or relationships.

Prevention and resolution strategies:

- Responsibilities (e.g. for delivering requirements) and resources shall be redistributed by a party in a senior position.
- Dependencies shall be dissolved by a party in a higher position.
- Another decision process shall be implemented.
- External pressure shall be redefined in a way that it does not influence the work or the requirements of the conflict party.

As most of the resolution strategies described here may not be within the responsibilities of the Requirements Engineer, he/she can only escalate structural conflicts and let other stakeholders intervene.

4.2.2.5 Relationship conflict

A relationship conflict may be based on negative experiences with the conflict parties in the past, or on other negative experiences from comparable situations with similar people. Often it is connected to emotions and miscommunication, which makes it a lot more difficult to solve.

Examples of relationship conflicts:

- Stakeholders verbally attack the other conflict party without good cause or in a highly emotional way. Facts and fair discussion seem to be irrelevant.
- In a discussion a stakeholder from the software department does not accept a requirement discussed with the system architect because she/he perceives architects as having no clue about the reality of programmers' work.
- A stakeholder does not accept the workshop invitation of the Requirements Engineer because his arch enemy will join the meeting.

If a relationship conflict is the reason for a discussion, one can observe that communication amongst the conflict parties is focused on the "relationship" of the "four-sides" model of Schulz von Thun [Schulz von Thun 1981]. Conflict parties may use the discussion of requirements to express their disagreement with the behavior of the other conflict party. Facts and fair discussion seem to be rather unimportant to the conflict parties.

Prevention and resolution strategies:

- Stakeholders should agree on meeting rules and procedures when emotions arise.
- The Requirements Engineer should elicit requirements from the involved conflict parties separately and without revealing the requirement sources. That way the elicitation activity remains more objective.
- Prevent the negative behavior of the conflict parties.
- Review and agreement shall be performed without direct involvement of the conflict parties or by a neutral third party.

As most of the resolution strategies described here do not lie in the responsibility of the Requirements Engineer, he/she can only escalate relationship conflicts and let other stakeholders intervene.

4.3 Conflict resolution

A prerequisite for the selection of a proper resolution technique is an in-depth understanding of the nature of the requirements conflict.

Based on this analysis and the project constraints, the Requirements Engineer can select a suitable negotiation technique. All the techniques described in this section are structured in a similar way to the elicitation techniques in Chapter 3.

The following general resolution techniques can be distinguished (see [IREB2017]):

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

4.3.1 Agreement

What is it?

Agreement is the result of a discussion in which an existing set of requirements, or a conflict resolution, is selected without change from a number of available options. To achieve such a resolution, it is essential that there is enough time to understand the positions of all stakeholders completely and convince them that the selected option is also their preferred choice.

Role of participants

The moderator can take the role of reminding stakeholders to converse in a constructive and efficient way. He shall be neutral to the options and be accepted in this role by the group. The involved stakeholders should concentrate on facts and allow others to ask questions.

Preparation

For a very big group it is advisable to name a representative per group. The stakeholders shall be invited, informed about the agenda and their expected contribution to the process.

Application

- The moderator shall define the topic and set an agenda with a timetable.
- The moderator shall present the data about the conflict and explain possible resolution techniques.
- The stakeholders shall present their arguments without interruptions, after which other parties can ask questions.
- The moderator follows the discussion and suggests a proper resolution. If this seems to be impossible because of the type of subject matter, type of conflict or because new conflicts arise, the moderator should end the discussion and suggest another resolution technique.

Result processing

A participant can summarize the result for all participants. The moderator should facilitate an informal agreement from all parties right away. If the options are too complex, the result and the affected requirements are completed later and reviewed individually.

Typical artefacts

Arguments for and against the different alternatives; the selected solution (including the requirements agreed by the conflict parties).

Chances

With agreement there is an opportunity for the conflict parties to understand each other and their requirements better. A positive outcome can provide additional motivation for the group and the result has a good chance of being long lasting.

Challenges

The challenges lie in presenting the essence of the conflict so that everybody knows what shall be discussed. For the moderator, it can be challenging to prevent the culture of discussion from negatively influencing the result, and not reaching agreement although an agreement seemed to be likely. It is challenging to keep the stakeholders on a factual level, to share discussion time equally and to stay within the timeframe.

Variants

Not applicable.

4.3.2 Compromise

What is it?

Compromise is the result of a discussion in which aspects of an existing set of requirements or conflict resolutions, as well as new aspects are combined to create new options. To achieve such a resolution, it is essential that there is enough time to reach a good understanding of the respective positions and all the aspects of the problem in order to negotiate a resolution that meets everyone's needs.

Role of participants

The moderator can take the role of reminding stakeholders to keep to the agreed rules and to continue discussing in a constructive and efficient way. He should be neutral to the options and be accepted in that role by the group. The involved stakeholders should concentrate on facts and allow others to ask questions in order to fully understand all aspects important to the conflict parties.

Preparation

Agreement should first have been considered and ruled out as an immediate option. For a very big group it is advisable to name a representative per group. The stakeholders shall be invited, informed about the agenda and their expected contribution to the process.

Application

- The moderator shall define the topic and set an agenda with a timetable.
- The moderator shall present the data about the conflict and explain possible resolution techniques.
- The stakeholders shall present their arguments without interruptions, after which other parties can ask questions.
- The moderator shall advise the conflict parties to negotiate for their most important aspects first.
- As soon as the conflict parties agree on a particular aspect the moderator should ensure that this agreement is documented.
- The moderator follows the discussion and encourages application of communication rules.

Result processing

A participant can summarize the result for all participants. The moderator should facilitate an informal agreement from all parties right away. If the options are too complex, the result (and the affected requirements) can be completed later and reviewed individually.

Typical artefacts

Argumentation as to why an agreement was not a suitable resolution technique. Different aspects discussed including the requirements the conflict parties agreed on.

Chances

With a compromise there is an opportunity for the conflict parties to understand each other and their requirements better without the need for complete agreement.

New aspects can be integrated and more stakeholders contribute to the created resolution. A positive outcome can provide additional motivation for the group and the result has a good chance of being long lasting.

Challenges

The challenges lie in presenting the essence of the conflict so that everybody knows what should be discussed. For the moderator, it can be challenging to prevent the culture of discussion from negatively influencing the result, as, for example, when the dominant party achieves a better result for itself but not for the whole project. The moderator should therefore ensure that stakeholders stay on the factual level, take an equal share of discussion time and keep to the specified timeframe.

Variants

Not applicable.

4.3.3 Voting

What is it?

Voting is the result of a selection from an existing set of requirements. It is critical that the possible options, or the set of requirements, are well understood by the decision makers. In order that dependencies or an imbalance of power should not influence the result, voting for an option should be secret.

Role of participants

Decision makers should understand the consequences of their choices and the available options. A moderator can lead them though the process and communicate the steps of the voting procedure, the available options and the result of the voting.

Preparation

Stakeholders with the authority to decide about the subject matter have to transfer the power for the decision to the selected stakeholders and agree to accept the outcome. In order to make clear what the stakeholders shall vote for, proposals for each option should be created. A neutral voting committee shall be selected and voting sheets with appropriate choices prepared.

Application

- Once it has been agreed to solve a conflict via the resolution technique “Voting”, each party involved in the conflict should prepare a proposal that describes their position and the possible consequences of the solution proposed to the decision makers. After the proposals are prepared and distributed the voting meeting can take place.
- In the voting meeting the moderator should explain the voting procedure and the respective positions to the decision makers.
- Optionally each party can present their proposal and answer the decision-makers’ questions.
- Voting sheets are collected and evaluated.

Result processing

The results of the voting should be documented.

Typical artefacts

Proposals for the suggested options including pros and cons, as well as the completed voting sheets.

Chances

Voting can accommodate the involvement of many stakeholders, which in turn can achieve greater stakeholder satisfaction.

Challenges

It can be difficult to explain complex options in such a way that they can be understood by every decision maker. Political trends could influence the decision.

Variants

For simple decisions, voting can be performed spontaneously without any preparation and finished very quickly. Various tools support decision making and support the involvement of stakeholders located anywhere in the world. Such tools may also support a voting procedure that is not synchronous.

4.3.4 Definition of variants

What is it?

Definition of variants is the result of the integration of the relevant, differing requirements into one solution in which the system can be configured to support either option. The configurator or the user of the system can then select a feature representing the desired set of requirements during configuration or at runtime. Often there is an additional set of requirements that have to be implemented in order to support the switching between the different options. As every option – as well as the switch between the options - has to be elicited and maintained, the definition of variants could turn out to be costly and result in complicated systems with rarely-used features.

Hint 4.3.1:

The Requirements Engineer has to assess whether the definition of variants is not in fact an escape from a proper conflict resolution process, and is indeed worth the additional effort.

The configuration of the system shall be implemented in such a way that the variants can be used independently and the conflict is really solved.

Role of participants

The Requirements Engineer should underline the differences between the variants so it is clear why a single solution option could not be achieved.

Preparation

Techniques like agreement, compromise and voting should first have been considered. All parties should agree that creating a variant causing additional effort in the future is the only possible resolution.

Application

- There could be separate meetings organized with each party to elicit the affected requirements for the different options, so that they do not interfere with each other.

- After the separate options are specified, the parties shall review the options for greater alignment. If there is a greater alignment, the common parts of the various options should then be combined.
- Where no further alignment is possible, the requirements for the switch between the variants should be specified.

Result processing

The results of the definition of variants should be documented.

Typical artefacts

Ideally the artefacts are the affected requirements for each option and the requirements for the mechanism to select the options.

Chances

With definition of variants the integration of different stakeholders' particular needs can be achieved. This can improve the level of involvement and increase stakeholder satisfaction.

Challenges

It can be difficult to create options that are really necessary and that minimize additional effort in the development process.

Variants

The definition of the deviation point may be very early or late in the product lifecycle. It can vary from configuration during runtime to configuration during setup to completely different products or product families. It is important to find the right deviation point in order to find a good balance between effort (mostly costs) and individual stakeholder needs.

4.3.5 Overruling

What is it?

Overruling is the result of the selection of existing set of requirements. It is critical that the possible options, or the set of requirements, are well understood by the decision maker. As the decision maker can overrule all involved conflict parties, the imbalance of power does not influence the result.

Role of participants

The decision maker should understand the consequences of his/her choices and the available options. He should also justify the decision so it can be accepted by all involved conflict parties. Experts can be consulted to collect arguments for the different options and to support the decision maker with the requisite knowledge.

Preparation

A decision maker should be identified who is accepted by all the conflict parties involved. Conflict parties or experts prepare proposals containing an explanation of all the options subject to the overruling.

Application

- The decision maker has to read the proposals and can ask questions.
- Optionally every party can present their proposal in person and answer the decision maker's questions.
- The decision maker announces the decision.

Result processing

The results including the reasons shall be documented.

Typical artefacts

A document containing the proposals and the selected option, as well as the reasoning for the decision.

Chances

The hierarchy in the organization may be used to find a resolution if there is no other way to find it.

Challenges

It can be challenging to create a proposal that contains all the facts, needs and consequences of the different options.

Variants

Overruling can be performed by a committee of decision makers. Rather than the manager of the conflict parties, a neutral expert who is accepted to all parties may take the decision instead.

4.3.6 Auxiliary techniques

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

- Non-violent communication [Rosenberg2015]
- Negotiation techniques [FiUP2012]
- Consider-all-facts [DeBono2006]
- Plus-minus-interesting [DeBono2006]
- Decision matrix [BiAB2006] [IsNe2013]

4.3.7 Finding a suitable conflict resolution technique

Based on the characteristics of a conflict, suitable negotiation techniques should be selected.

	Agreement	Compromise	Definition of variants (configurability)	Voting	Overruling	+ recommended 0 applicable, decide on other constraints whether to use it - not recommended
1. High number of stakeholders and/or different opinions	-	-	0	+	+	Listening to all arguments could take too much time and would be hard to achieve.
2. High criticality of the situation[11]	+	-	-	-	0	Decisions shall be well thought through and accepted by all stakeholders.
3. Wide distribution of the stakeholders	-	-	+	+	+	See 1. It could be smart to create individual variants to serve different (local) needs. Voting and overruling can be done virtually.
4. High time pressure for conflict resolution	-	-	-	+	+	Defining variants will take too much time, as two variants and the configuration mechanism have to be defined.
5. Clarity of the result is important	+	+	-	0	+	By defining a variant it is not easy to explain what the product contains.
6. Low social competence of the stakeholders[12]	-	-	+	+	+	Agreement and Compromise require social competence in order to exchange opinions.
7. Complicated situation[13]	-	+	-	-	0	In complicated situations a conflict party has to concentrate on aspects they can understand. Combining competencies and accept different aspects leads to compromises. In case an expert with the necessary competency is the decision maker, overruling is possible.
8. Long lifetime of the results	+	-	+	-	-	Bad compromises, voting and overruling can lead to conflict parties that will not accept the result and want to change it over time.

[11] Means the wrong result of the resolution leads to a high risk for the implementation.
[12] Stakeholders are not able or willing to listen to each other and are not capable to accept other opinions.
[13] The subject matter is that complicated than not all stakeholders can understand the consequences.

	Agreement	**Compromise**	**Definition of variants** (configurability)	**Voting**	**Overruling**	**+ recommended** **0 applicable, decide on other constraints whether to use it** **- not recommended**
9. Low motivation of the stakeholders (to take part actively)	-	-	+	+	+	Agreement and compromise require some time to discussions and involvement, so high motivation is needed.
10. Poor time availability of the stakeholders	-	-	0	+	+	Agreement and compromise require some time for discussions and participation. Depending on the nature of the variant, it can be less time-consuming to define only his own variant instead of the whole system.
11. Conflict concerning data	+	+	0	+	-	Depending on the content, variants can be a proper technique.
12. Conflict concerning interests	-	0	+	0	+	Voting does not really solve the conflict, as the majority always wins.
13. Conflict concerning value	-	-	+	-	0	As values cannot be changed in the short term, creating variants is the only way to satisfy all values. If creating variants is not a suitable technique, overruling is the only option.
14. Conflict concerning structural	-	0	-	+	+	The negative effects of the structural conflict prevent an honest agreement and a fair discussion which make agreement or compromise unlikely. However, compromise could be the only remaining resolution technique. As voting is anonymous and overruling involves an instance above both conflict parties, distribution of power is minimized. If the conflict in the distribution of majority voting is not a suitable technique.
15. Conflict concerning relationship	-	-	-	-	-	There is no suitable technique or the responsibility to solve this type of conflict does not lie with Requirements Engineering.

4.4 Documentation of conflict resolution

After its resolution, the conflict should be properly documented. Apart from the characteristics of the conflict mentioned in Section 4.2, this should include in particular:

- Assumptions concerning the conflict and its resolution
- Constraints influencing the choice of conflict resolution technique and/or resolution
- Potential alternatives considered
- Conflict resolution, including reasons for the chosen resolution
- Decision-makers and other contributors

If not documented, stakeholders may simply forget or ignore the decisions that have been taken, or try to change them afterwards. This often occurs in situations where the requirements conflict itself is resolved, but an underlying social conflict remains unresolved.

5. Skills of the Requirements Engineer

This chapter gives an overview of important skills for the Requirements Engineer. Since communication is a core element of Requirements Engineering, important communication theories will be presented to provide a deeper understanding of the mechanics that drive communication. Finally, as personal development is an ongoing process, the final three sections of this chapter deal with self-reflection, opportunities for personal development and lifelong learning.

5.1 Required skills in the areas of elicitation

The approaches and techniques explained in the previous chapters form the basic skill set of each Requirements Engineer. Mastering these (hard) skills is a prerequisite for operating at an advanced level. But that is not enough: to be successful, the Requirements Engineer must also possess a number of soft skills (see, e.g., [Klaus2007]).

The IREB CPRE Foundation Level syllabus [IREB2017] mentions communication skills, analytical thinking, empathy, conflict resolution skills, moderation skills, self-confidence and the ability to convince.

Marcel Robles [Robles2012] gives an overview of the top ten most important soft skills:

- Communication – oral, speaking capability, written, presenting, listening
- Courtesy – manners, etiquette, business etiquette, gracious, says please and thank you, respectful
- Flexibility – adaptability, willing to change, lifelong learner, accepts new things, adjusts, teachable
- Integrity – honest, ethical, high morals, has personal values, does what's right
- Interpersonal Skills – nice, personable, sense of humor, friendly, nurturing, empathetic, has self-control, patient, sociability, warmth, social skills
- Positive Attitude – optimistic, enthusiastic, encouraging, happy, confident
- Professionalism – businesslike, well-dressed, appearance, poised
- Responsibility – accountable, reliable, gets the job done, resourceful, self-disciplined, wants to do well, conscientious, common sense
- Teamwork – cooperative, gets along with others, agreeable, supportive, helpful, collaborative
- Work Ethic – hard working, willing to work, loyal, initiative, self-motivated, on time, good attendance

While all of the above are relevant for every professional, the following characteristics are in addition particularly important in relation to requirements elicitation:

- Contextual awareness – knowing the context in which you are operating and adjusting your approach accordingly
- Ethical conscience – any technology can have a negative impact on people, society and the environment. If the Requirements Engineer detects such a negative impact, he should make it explicit to encourage discussion to mitigate the impact.
- Intercultural competency – able to work in and with different (business, domain, regional, etc.) cultures
- Leadership – able to lead stakeholders to a certain goal

- Motivating nature – able to inspire stakeholders for a certain goal
- Neutrality – able to serve all (relevant) stakeholders and their interests equally, without personal interest
- Reflection – able to receive feedback and evaluate situation and own behavior
- Self-awareness – knowing your own position in relation to the stakeholders and adjusting your approach accordingly

Depending on the business, the project context and the stakeholders at hand, certain skills may need more attention than others. Nonetheless, all are important: in fact, contextual awareness refers to the ability to apply the right skills at the right time, when a certain situation demands it.

Of all soft skills, communication skills are the key success factor for the Requirements Engineer – it is no coincidence that Robles mentions this skill first.

All interaction between the Requirements Engineer and stakeholders (being the prime sources of requirements) is a form of communication and most, if not all, of the aforementioned skills play a role. Some skills, for example integrity and neutrality, are important for the communication itself; others, like leadership and teamwork, can only be realized through communication.

5.2 Communication theory and communication models

Communication can be seen as a way for individuals to exchange messages and to create meaning. It includes any behavior of a person with the purpose of making something clear to another person who perceives and interprets it. Moreover, communication is considered a complex, interpersonal process that uses any combination of speech, writing and other signals as a basis for the exchange of concepts, thoughts, opinions and information.

Communication is effective if the exchange results in congruence between the intended and the perceived meaning. Effective communication is not necessarily efficient in the sense that the desired outcome is achieved with a minimum of effort, time, complexity, and investment of resources. In fact, as good communication heavily relies on redundancy, more efficiency may easily lead to less effectiveness. The Requirements Engineer should make sure that communication with stakeholders is effective within reasonable limits of efficiency.

Communication is often goal-directed: the sender has the intention of invoking certain behavior at the receiver side. In elicitation and conflict resolution, the Requirements Engineer is trying to motivate stakeholders to behave in an open and collaborative way and to reveal all relevant information freely.

To a large extent, successful elicitation and conflict resolution depends on the proper understanding of the "nuts and bolts" of communication. Below, several models from the field of communication theory are presented with their relevance for requirements elicitation.

The Shannon-Weaver model [ShWe1971] is often considered to be the "mother of all communication models". It concentrates on *encoding* a *message* from a *sender* to a *receiver*, who *decodes* it after its *transmission* through a certain *channel* with the risk of *noise* disturbing this message (see Figure 25).

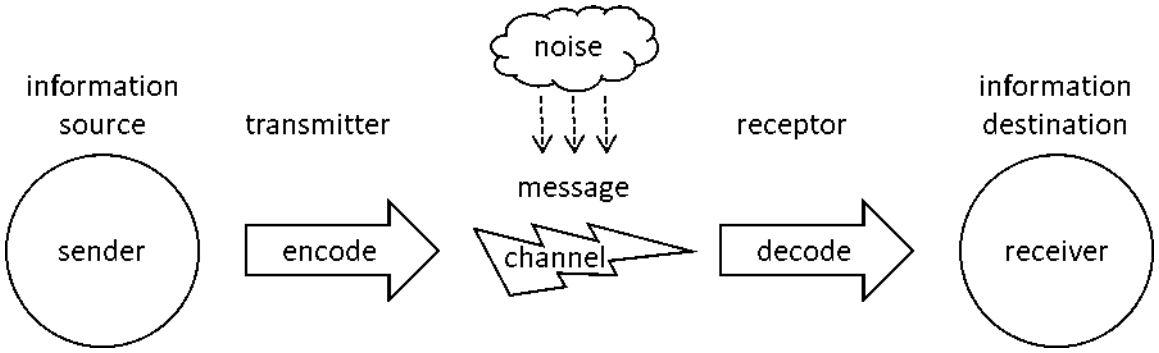

Figure 25: The Shannon-Weaver communication model

In Requirements Engineering, it is often "virtual" noise that needs attention, as this may lead to incorrect requirements and conflicts: biases, gossip, hear-say, hidden agendas. The Requirements Engineer should be able to recognize this kind of noise, check the compatibility of encoding and decoding ("Do we understand each other?"), and select the proper channels (e.g., meetings, presentations, publications) for communication with the stakeholders.

Schramm [Schramm1971] contributed two models to communication theory.

Communication is seen as a *social interaction* between a sender and a receiver. Schramm's first model (see Figure 26) makes clear that a message can only be successfully communicated if it fits into an area of shared experience. Schramm also indicated that we should examine the impact (both desired and undesired) that a message has on the target of the message.

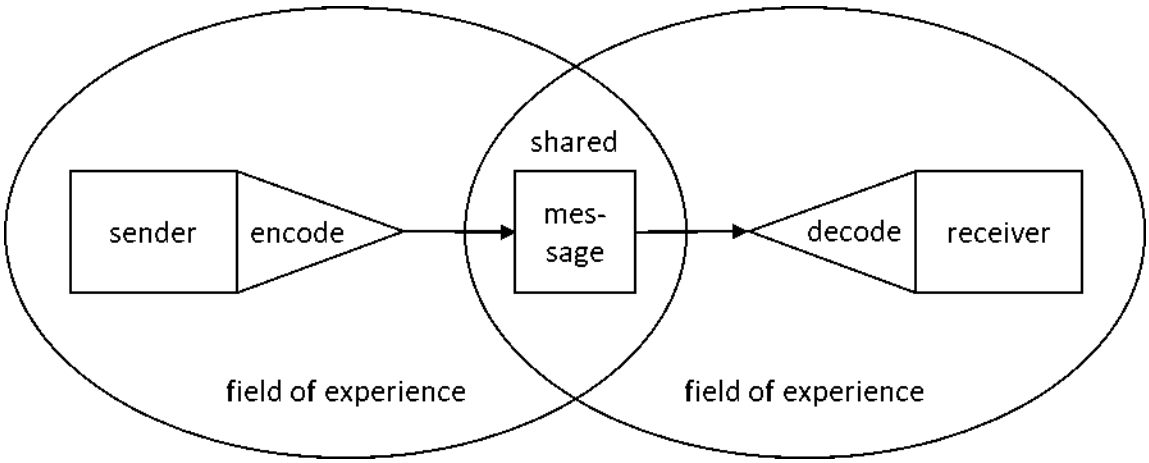

Figure 26: Schramm's communication model on shared areas of experience

The Requirements Engineer should check if the necessary sharing exists in the relationship with the stakeholders. A lack of it may lead to failure. The Requirements Engineer can enlarge the area of shared experience, for example by acquiring domain expertise through self-study or by giving training on requirements issues to the stakeholders.

Schramm also developed the *circular model* of communication (see Figure 27). In this model, the sender encodes a message which is decoded and interpreted by the receiver, who then responds by encoding another message and passing it along. In elicitation, this pattern can be observed, for example when the Requirements Engineer asks questions to a stakeholder, listens to the answers and summarizes them afterwards. The idea of feedback fits into this model.

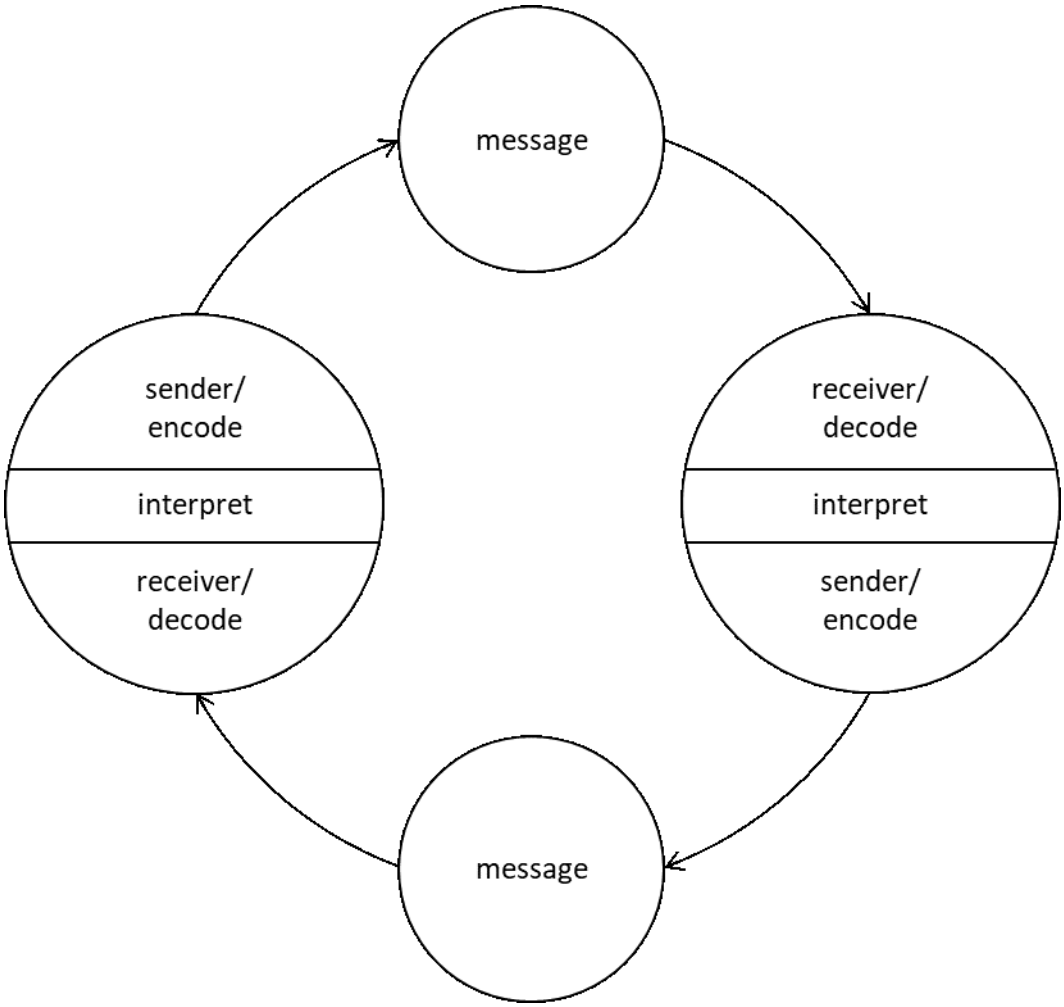

Figure 27: Schramm's circular communication model

The 4-sides communication model of Schulz von Thun [ScTh1981] puts the *message* at the center and describes *four aspects* to be considered (see Figure 28).

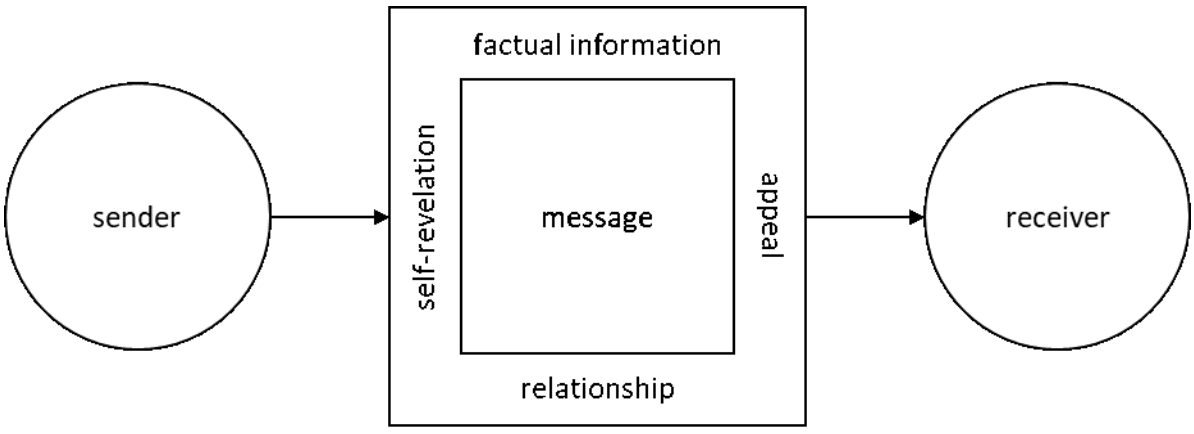

Figure 28: Communication model of Schulz von Thun

The Requirements Engineer must be aware of all these aspects:

- *Factual information*: the factual content of a message is the basis for eliciting and consolidating requirements.
- *Self-revelation*: this relates to the extent to which a stakeholder is committed to a certain requirement, whereas the Requirements Engineer should maintain strict neutrality.

- *Relationship*: the Requirements Engineer should seek an open and constructive relationship with the stakeholders and should verify the perspective of the stakeholders.
- *Appeal*: the appeal aspect can give clues on what the stakeholder expects from the Requirements Engineer and provides feedback about the factual content communicated.

All these models contribute to the understanding of communication between the Requirements Engineer and the stakeholders. A common idea is that communication is about *sharing meaningful concepts* between individuals. The key point is that during this communication, information may be lost, added, distorted or misinterpreted. These so-called *transformational effects (see 3.3.3)* result in differently perceived concepts amongst the individuals concerned.

The Requirements Engineer should assure the quality of his communication by mitigating these effects. It is essential to pay proper attention to the *encoding* and *decoding* of messages, choosing the right *channels*, avoiding *noise*, and checking the correct interpretation through *feedback*.

Redundancy is the primary solution for communication problems. Usually, this means that the same information is transmitted several times, often through different channels. Examples are a diagram accompanied by an explanation or a presentation supported by non-verbal gestures. If, and only if, redundant messages are consistent do they support the quality of communication.

Hint 5.2.1:

- Use redundancy in all your communication.
- Always ask for feedback.
- Watch out for transformational effects.
- Try to inspire the stakeholder to rephrase suspect statements with other words, concrete examples or through other media e.g. drawings, metaphors.

5.3 Self-reflection on personal skills in requirements elicitation

"I know that I know nothing" (Socrates)

"I know nothing, I'm from Barcelona" (Manuel)

The syllabus, this handbook, and any accompanying training lay out the foundation for the successful application of the presented methods and techniques. However, the development and improvement of personal skills for the elicitation of requirements is a long-term learning process.

The basis for improvement is self-reflection. There are three types of reflection:

1. Thinking about what you have planned to do and critically re-thinking that plan: Could it be done differently? What do I expect from the planned activities? What do I want to learn? Can I improve that way? (prospective reflection)
2. Thinking about how I am currently doing my job: Do I behave professionally? Do I apply what I have learned? What are my current fears and hopes regarding the elicitation and conflict resolution activities I am involved in? (accompanying reflection)
3. Thinking about what you did and how you performed your elicitation and conflict resolution tasks in the past (retrospective reflection).

Figure 29: Three types of reflection

Even if the Requirements Engineering during a development project is considered a success, there are typically several opportunities for improvement.

Hint 5.3.1:

Here is a short checklist of questions to help you start your self-reflection process:

- Has a technique delivered the expected results / contributed to the development? If yes, what was the main driver of the success? If no, what was the reason for the failure? What could I have done differently? What should I improve next time?
- Did the stakeholder(s) accept the elicitation / conflict resolution techniques applied? If yes, what was the reason for them to collaborate? If no, what was the problem? What could I have done differently? What should I improve next time?
- Was the effort for a technique justifiable with respect to the contribution to the development? If not, why? Was it a problem of the technique in our context (i.e. I should have selected a more appropriate technique) or how did I fail in applying the technique appropriately?
- Which technique might have allowed eliciting requirements that came up late in the development at an earlier time? What was the reason for not identifying them earlier?
- Which alternative techniques might also have been applied? Would it be beneficial to consider specific elicitation or conflict resolution techniques for a future project in a similar situation? Is it necessary to acquire further knowledge and experience on them (see Section 5.4)?

As an advanced Requirements Engineer, having done many self-reflection sessions, you will develop more and more questions, will focus on specific aspects, etc. until you end up with your own personalized checklist. Or – if you have become an expert in self-reflection – you may identify the topics to reflect in the course of your work. However, as a beginner (inexperienced in self-reflection), beware of over-simplification: laying on the bed, pretending to think about yourself and doing professional self-reflection might result in a relaxing afternoon nap, rather than on insights into yourself and your elicitation proficiency!

Hint 5.3.2:

Therefore, here are some reflection hints for novices:

- Start your self-reflection activity in front of a nice cup of coffee or tea (or whichever drink you prefer); have your notes and input data available (see Hint 5.3.3).
- Plan your self-reflection process: What question do you first want to think about? Initially use an existing checklist of reflection-questions such as the one provided in Hint 5.3.1.
- Take notes on your thinking outcome.
- Stick to the selected topic! Avoid changing the topic before you have addressed all aspects of it. Systematically consider every aspect of the selected question.
- Do not just think about problems and mistakes and how to avoid these in the future; also capture criteria and lessons learned relating to successes/positive outcomes.
- Relate your thinking to known concepts and established theories. If necessary, state that you need to learn more about a specific technique and method afterwards. It is also possible to look at a text book (or the Internet) to assess a performed elicitation or conflict resolution action. Make sure not to get lost in the details of a written source. Quickly return to the reflection question that was the reason for the excursion into the literature.

Hint 5.3.3:

As input for your self-reflection, you might first collect data about you and your professional behavior:

- If possible, take a video or voice recording of an elicitation activity you perform (e.g. an interview).
- Ask your peers for feedback on specific aspects of your professional performance. For example, a colleague might observe you interviewing a stakeholder or moderating a workshop. Or an expert analyzes your observation notes or assesses a questionnaire that you have created. Reading the findings and discussing them with the observer/assessor might provide valuable input for your self-reflection.
- Ask your stakeholders for feedback. This could be done with a brief interview (for example immediately after an elicitation interview) or with a short questionnaire. However, be careful not to mix elicitation and conflict resolution activities with asking for feedback on your behavior. In the case of conflicts or reluctance to provide information on the elicitation topic, stakeholders might not be prepared to support your self-reflection process.
- Apply a human-centered approach to your own work: for example, testing a questionnaire that you have developed with a few preliminary stakeholders may provide insights that not only improve the questionnaire itself, but also serve as input for reflection on your performance.
- Ask a senior Requirements Engineer to be your coach.
- Apply the well-known diary technique to your own job. Taking notes on problems, successful actions, informal feedback, lessons learned, etc. helps you to remember when performing the reflection process. An assessment sheet of previously defined capabilities is also a suitable measuring instrument [SmMa2011].

Collecting input from people around you is sometimes called 360° feedback [LeLu2009].

5.4 Opportunities for personal development

Insufficient practical experience is very often presented as a reason for not applying a specific elicitation or conflict resolution technique. Such an attitude might be understandable in terms of project success (the Requirements Engineer applies the techniques he knows best to ensure project success); in terms of personal development, however, this attitude is not helpful as the Requirements Engineer will never learn unfamiliar techniques and thus extend his tool box. In the following section we present two good practices that allow the application of new techniques in running projects: application in low-risk settings and application in parallel to a familiar technique.

An unfamiliar technique can be applied in a low-risk setting to minimize the negative effects in case the technique does provide the expected outcome. What constitutes a low-risk setting depends, of course, on the project context. Typical characteristics of a low-risk setting might be:

- Applying a technique to only a reduced subgroup of stakeholders. For example, the Requirements Engineer applies apprenticing only to a small number of stakeholders.
- Applying a technique for a limited timeframe. For example, the Requirements Engineer plans a very short field observation (e.g. one hour) with certain stakeholders.
- Applying a technique in a friendly environment. For example, a Requirements Engineer who has worked for a long time in a project and has established a good relationship with his stakeholders. In such an environment, the application of unfamiliar techniques is typically easier.

- Applying a technique to an aspect of the system that is not considered critical. For example, editing the user profile in a web shop might be considered as an uncritical feature. Therefore, the Requirements Engineer might use an unfamiliar technique to elicit requirements for such a feature.

Hint 5.4.1:

If you apply an unfamiliar technique, be honest with your stakeholders. Make clear to them that the applied technique is new to you and ask them for feedback afterwards. This honest approach with stakeholders reduces the pressure on you. Stakeholders typically recognize if you are uncertain with a new technique and are often willing to give you feedback to improve your skills in that area.

It is further possible to apply an unfamiliar technique in parallel to a familiar technique to reduce the risk of failure. Parallel application may take place in various forms:

- Synchronous parallelism: In case a large group of stakeholders must be addressed during requirements elicitation, a smaller subgroup of these stakeholders can apply an unfamiliar technique. For example, 30 stakeholders should participate in a creativity workshop. A subgroup of 6 stakeholders might be invited to perform an unfamiliar technique (e.g. 6 thinking hats), whereas the majority applies a familiar technique (e.g. brainstorming).
- Asynchronous parallelism: If it is not possible (or desirable) to split a group of stakeholders to perform two techniques in parallel, two techniques might instead be performed sequentially. For example, five end users of a new system are to participate in requirements elicitation by means of interviews. In such a situation, the Requirements Engineer might plan for an additional field observation right after each interview.

Hint 5.4.2:

If you apply an unfamiliar technique in parallel to a familiar technique, plan for an explicit comparison of the results of both techniques with your stakeholders. In this way your stakeholders may also benefit from the new results.

5.5 Learning from previous experience – lifelong learning

The essential components of a personal training process that fosters learning from previous experience are:

- *Regular measurement of your own ability profile*: An awareness of your own strengths and weaknesses in terms of the skills profile is the basis for successful further development. Regular analyses (for example through self-assessment questionnaires or conversations with customers or colleagues) of your own skill profile promote an awareness of your own strengths and weaknesses.
- *Training measures*: In order to improve your skills profile, further education, training or coaching in one or more elicitation/conflict resolution techniques or the required skills (e.g., leadership or motivational training) should be carried out.

- *Improvement in everyday work*: Dedicated training measures are a first step to improving your own skills. Substantial progress is, however, only achieved through application and practice in everyday work. Continuous improvement of your skills must therefore be an integral part of your practical work. Good progress can be achieved if improvement in individual skills is promoted (e.g. improving leadership skills in workshops) over an extended period (at least 4 weeks).
- *Mentoring measures*: An alternative to intensive training is learning a technique or method from an experienced mentor. Typically, you can first assist your mentor during the application of a technique and learn from observing their behavior. Later on, your mentor delegates responsibilities to you, observes your performance and provides feedback.

Hint 5.5.1:

Develop and cultivate your personal path of continuous improvement for Requirements Engineering skills. Learning the facts of most techniques is easy; developing excellence in the application of these techniques is another story. There is no predefined stairway to heaven (or RE excellence)! Use the elements described in this Section as a toolbox for finding your personal improvement path.

6. References and further reading

[A4qu2018] Alliance for Qualification: A4Q Design Thinking Foundation Syllabus. https://isqi.org/en/a4q-design-thinking-foundation-level, 2018. Last visited February 2019.

[AlIS1977] C. Alexander, S. Ishikawa, M. Silverstein: A Pattern Language - Towns - Buildings - Construction. Oxford University Press, New York, 1977.

[Alexander2005] I. F. Alexander: A Taxonomy of Stakeholders - Human Roles in System Development. International Journal of Technology and Human Interaction, Vol 1, 1, 2005, pages 23-59.

[BaGL1996] V.R. Basili, S. Green, O. Laitenberger, et al.: The Empirical Investigation of Perspective-Based Reading. Empirical Software Engineering 1: 133. https://doi.org/10.1007/BF00368702, 1996. Last visited February 2019.

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

[BaGr2005] R. Bandler, J. Grinder: The Structure of Magic, Vol. 1 - A Book about Language and Therapy; 1st edition, 2005.

[BeHo1998] H. Beyer, K. Holtzblatt: Contextual Design - Defining Customer-Centered Systems. Morgan Kaufmann, 1998.

[Beveridge 1957] W. I. Beveridge.: The Art of Scientific Investigation. The Blackburn Press, 1957.

[Design Council 2007] Design Council: 11 lessons: managing design in 11 global brands – A study of the design process. https://www.designcouncil.org.uk/resources/report/11-lessons-managing-design-global-brands, 2007. Last visited February 2019.

[BiAB2006] Stefan Biffl, Aybuke Aurum, Barry Boehm: Value-Based Software Engineering. Springer-Verlag, Berlin, 2006.

[Bitkom2017] Bitkom: Role Mode Digital Design, https://www.bitkom.org/sites/default/files/pdf/noindex/Publikationen/2017/Leitfaden/20171208-role-model-digital-design.pdf. Last visited February 2019.

[Boehm2006] B. Boehm: A View of 20th and 21st Century Software Engineering. 28th international conference on Software engineering (ICSE '06), p. 12-29, 2006.

[Bourne2015] L. Bourne: Making Projects Work: Effective Stakeholder and Communication Management. CRC Press, 2015.

[BrIs2005] J. Brown, D. Isaacs: The World Café: Shaping Our Futures Through Conversations That Matter, Berrett-Koehler Publishers, 2005.

[Brown2009] T. Brown: Change by Design – How Design Thinking Transforms Organizations and Inspires Innovation. HarperCollins, 2009.

[BüHe2015] S. Bühne, A. Herrmann: Handbuch Requirements Management nach IREB Standard (version 1.0.1), IREB e.V., Karlsruhe, 2015. https://www.ireb.org/en/downloads/#cpre-advanced-level-requirements-management-handbook. Last visited April 2019.

[Buxton2007] B. Buxton: Sketching User Experiences – Getting the design right and the right design. Morgan Kaufmann, San Francisco, 2007.

[BuBu2005] T. Buzan, B. Buzan: Das Mind-Map-Buch – Die beste Methode zur Steigerung Ihres geistigen Potenzials, Morgan Kaufmann; 5. aktualisierte Auflage, 2005.

[Cambridge2017] Cambridge Dictionary https://dictionary.cambridge.org/dictionary/english/abstraction. Last visited February 2019.

[Carrol2003] JM. Carrol: Making Use. The MIT Press, 2003.

[Chernak2012] Y. Chernak: Requirements reuse: the state of the practice. In: Proceedings of the 2012 IEEE international conference on software science, technology & engineering (SwSTE), p. 46–53

[Cockburn2001] A. Cockburn: Writing Effective Use Cases. Addison-Wesley, 2001.

[CRCN2014] A. Cooper, R. Reimann, D. Cronin, C. Noessel: About Face: The Essentials of Interaction Desig. 4th Edition, John Wiley & Sons, Inc, Hoboken, 2014.

[CoSh2007] T. Colburn, G. Shute: Abstraction in Computer Science, Minds & Machines, Vol. 17, p. 169–184, 2007.

[Cooper2004] A. Cooper: The Inmates Are Running the Asylum-Why High-tech Products Drive Us Crazy and How to Restore the Sanity. Que, Indianapolis, 2004.

[Couger1996] J. D. Couger: Creativity and Innovation in Information Systems Organizations. Byod & Fraser, 1996.

[CrOB2006] O. Creighton, M. Ott, B. Bruegge: Software Cinema: Video-based Requirements Engineering. 14th IEEE International Requirements Engineering Conference (RE'06), 2006.

[CHQW2016] T. Cziharz, P. Hruschka, S. Queins, T. Weyer: Handbook of Requirements Modeling According to the IREB Standard (version 1.3), IREB e.V., Karlsruhe, 2016. https://www.ireb.org/en/downloads/#handbook-cpre-advanced-level-requirements-modeling

[DeBono2006] E. DeBono: Edward DeBono's Thinking Course - Powerful Tools to Transform Your Thinking. BBC Active, London, 2006.

[DeDe2011] K. Dewalt, B. Dewalt: Participant observation - A guide for fieldworkers. 2nd Edition. AltaMitra Press, Plymouth, UK, 2011.

[ElOs2017] R. Elamin, R. Osman: Towards Requirements Reuse by Implementing Traceability in Agile Development, in: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC) Italy, http://ieeexplore.ieee.org/abstract/document/8029969/?part=1, 2017. Last visited February 2019.

[FiUP2012] R. Fisher, W. Ury, B. Patton: Getting to Yes - Negotiating an agreement without giving in, 3rd rev. ed. Random House Business, New York, 2012.

[GHJV1994] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns. Elements of Reusable Object-Oriented Software. Prentice Hall, 1994.

[Glasl1999] F. Glasl: Confronting Conflict - A first-aid kit for handling conflict. Hawthorn Press, Gloucestershire, 1999.

[Glasl2004] F. Glasl: Konfliktmanagement: Ein Handbuch für Führungskräfte, Beraterinnen und Berater. 11th ed., Freies Geistesleben, Stuttgart, 2004.

[GoBe2015] L. Goldin, D. M. Berry: Reuse of requirements reduced time to market at one industrial shop: a case study, Requirements Engineering Journal 2013. In: Requirements Engineering-Volume 20, Issue 1, March 2015.

[Goodwin2009] K. Goodwin: Designing for the Digital Age. Wiley, New York, 2009.

[Gothelf2013] J. Gothelf: Lean UX – Applying Lean Principles to Improve User Experience. O'Reilly, Sebastopol, 2013.

[Gottesdiener2002] E. Gottesdiener: Requirements by Collaboration: Workshops for Defining Needs, Addison-Wesley Professional, 2002.

[GoWo2005] T. Gorschek, C. Wohlin: Requirements Abstraction Model, Requirements Engineering Journal 2005. In Requirements Engineering Volume 11, Issue 1, Pages 79 – 101, December 2005.

[Groen et al.2017] E. C. Groen, N. Seyff, R. Ali, F. Dalpiaz, J. Doerr, E. Guzman, M. Hosseini, J. Marco, M. Oriol, A. Perini, M. Stade: The Crowd in Requirements Engineering - The Landscape and Challenges, IEEE Software, vol. 34, no. 2, pp. 44-52, 2017.

[GrKo2016] E. C. Groen, M. Koch: How Requirements Engineering can benefit from crowds - Driving innovation with crowd-based techniques. Requirements Engineering Magazin Vol. 2. https://re-magazine.ireb.org/issues/2016-2-take-the-broader-view/how-requirements-engineering-can-benefit-from-crowds. 2016. Last visited February 2019.

[HaPy2012] R. Hartson, P.S. Pyla: The UX Book: Process and Guidelines for ensuring a Quality User Experience. Morgan Kaufmann, San Francisco, 2012.

[IREB2017] IREB e.V.: Syllabus CPRE Foundation Level, version 2.2. https://www.ireb.org/downloads/#syllabus-foundation-level, 2015. Last visited February 2019.

[IsNe2013] A. Ishizaka, P. Nemery: Multi-criteria decision analysis. Methods and software. John Wiley & Sons, Ltd, Chichester, 2013.

[ISO25010] ISO/IEC 25010:2011: Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. International Organization for Standardization, Geneva, 2011.

[ISO29148] ISO/IEC/IEEE29148: Systems and software engineering — Life cycle processes — Requirements Engineering. International Organization for Standardization, Geneva, 2011.

[ISO9241.11] ISO9241: Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 11: Guidance on usability. International Organization for Standardization, Geneva, 1998.

[ISO9241.210] ISO9241: Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems. International Organization for Standardization, Geneva, 2010 (replaces ISO 14407-210: 1999).

[JoDö2004] I. John, J. Dörr: Requirements Engineering, basierend auf existierenden Systemen, in: G. Böckle, P. Knauber, K Pohl, K Schmid. "Software-Produktlinien- Methoden, Einführung und Praxis", dpunkt.Verlag, Heidelberg 2004, p.153-163.

[KaBP2002] C. Kaner, J. Bach, B. Pettichord: Lessons Learned in Software Testing – A Context-Driven Approach. Wiley, New York, 2002.

[Katie2017] Katie: Pril Gets Pranked. Social Media for Business Performance, https://smbp.uwaterloo.ca/2017/05/pril-gets-pranked/, 2017. Last visited February 2019.

[Klaus2007] P. Klaus: The Hard Truth About Soft Skills - Workplace Lessons Smart People Wish They'd Learned Sooner. HarperCollins Publishers, New York, 2007.

[KnZK2016] J. Knapp, J. Zeratsky, B. Kowitz: Sprint - How to Solve Big Problems and Test New Ideas in Just Five Days. Simon & Schuster. 2016.

[Koelsch2016] G. Koelsch: Requirements Writing for System Engineering. Apress, 2016.

[Koes1964] A. Koestler: The Act of Creation. Last Century Media, 2014.

[Kumar2013] V. Kumar: 101 Design Methods – A Structured Approach for Driving Innovation in Your Organization. Wiley, 2013.

[Kvale2008] S. Kvale: Doing Interviews. SAGE, 2008.

[Lauenroth2014] K. Lauenroth: What does it mean to say „requirement“?-An inquiry into the abilities of the human mind and the meaning of the word „requirement“. Requirements Engineering Magazin Vol. 1. http://re-magazine.ireb.org/issues/2014-1-learning-to-fly/what-does-it-mean-to-say-requirement, 2014. Last visited February 2019.

[LeLL2018] M. Lewrick, P. Link, L. Leifer: The Design Thinking Playbook: Mindful Digital Transformation of Teams, Products, Services, Businesses and Ecosystems. Wiley, New Jersey, 2018.

[LeLu2009] R. Lepsinger, A.D. Lucia: The Art and Science of 360 Degree Feedback. 2nd ed., Wiley, San Francisco, 2009.

[LiHB2003] W. Lidwell, K. Holden, J. Butler: Universal Principles of Design. Rockport, 2003.

[LiFi2012] S. L. Lim, A. Finkelstein: StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation, in IEEE Transactions on Software Engineering, vol. 38, no. 3, pp. 707-735, 2012.

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

[LoSL2017] H. van Loenhoud, P. Steiger, K. Lauenroth: The goal is to solve the problem - Some thoughts on problems and goals in the context of Requirements Engineering. RE magazine 2017-02. https://re-magazine.ireb.org/articles/the-goal-is-to-solve-the-problem, 2017. Last visited February 2019.

[Maiden et al.2010] N. Maiden, S. Jones, I. Karlsen, R. Neill, K. Zachos, A. Milne: Requirements Engineering as Creative Problem Solving: A Research Agenda for Idea Finding. 18th IEEE International Requirements Engineering Conferenec (RE), 2010.

[MaGi2001] N. Maiden, A. Gizikis: Where Do Requirements Come From? IEEE Software Vol. 18, No. 5, 2001, S.10-12.

[McConnell2006] S. McConnell: Software Estimation – Demystifying the Black Art, Microsoft Press, 2006.

[McElroy2017] K. McElroy: Prototyping for Designers: Developing the Best Digital and Physical Products. O'Reilly, 2017.

[McGrBa2017] J. McGrath, B. Bates: The Little Book of Big Management Theories. 2nd Edition, Pearson Business, 2017.

[Miller2009] R.E. Miller: The Quest for Software Requirements. MavenMark Books, 2009.

[Moore2014] C. W. Moore: The Mediation Process – Practical Strategies for Resolving Conflicts. 4th ed., John Wiley & Sons, Hoboken, 2014.

[Osborn1948] A. F. Osborn: Your creative power: How to use imagination. C. Scribner's Sons, 1948. (Accessed as digital reprint: Read Books Ltd. (epub eBook), April 2013)

[Owen2008] H. Owen: Open Space Technology: A User's Guide. Berrett-Koehler Publishers, 3rd Ed., 2008.

[Dsch2012] d.School: An Introduction to Design Thinking PROCESS GUIDE. Hasso Plattner Institute of Design, Stanford, 2012.
https://dschool-old.stanford.edu/groups/designresources/wiki/36873. Last visited February 2019.

[Pohl2010] K. Pohl: Requirements Engineering – Foundations, Principles, and Techniques. Springer, 2010.

[PoRu2015] K. Pohl, C. Rupp: Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant, Rocky Nook, Santa Barbara, 2015.

[Portigal2013] S. Portigal: Interviewing Users: How to Uncover Compelling Insights. Rosenfeld Media, Brooklyn, 2013.

[Reinertsen1997] D. Reinertsen: Managing the Design Factory: A Product Developers Tool Kit. Free Press, 1997.

[RiFl2014] M. Richter, M. Flückiger: User-Centred Engineering – Creating Products for Humans. Springer, Berlin Heidelberg, 2014.

[Robertson2001] S. Ian Robertson: Problem Solving. Psychology Press, 2001.

[Robles2012] M.M. Robles: Executive Perceptions of the Top 10 Soft Skills Needed in Today's Workplace. Business Communication Quarterly 75(4) p. 453–465, 2012.

[RoRo2013] S. Robertson, J. Robertson: Mastering the Requirements Process: Getting Requirements Right. Third Edition, Pearson Education, London, 2013.

[Rosenberg2015] M. B. Rosenberg: Nonviolent Communication - A Language of Life. 3rdrev. ed., Puddle Dancer Press (US), Encinitas, 2015.

[RuCh2008] J. Rubin, D. Chisnell: Handbook of Usability Testing- How to Plan, Design, and Conduct Effective Tests.Wiley; Idianapolis, 2008.

[Rupp et al.2014] C. Rupp, die SOPHISTen: Requirements-Engineering und –Management - Aus der Praxis von klassisch bis agil. 6th ed., Carl Hanser Verlag, München, 2014. (selected chapters English version see http://www.sophist.de/en/infopool/downloads. Last visited February 2019.)

[Schramm1971] W.L. Schramm: How communication works, in W.L. Schramm, ed., (1971). The Process and Effects of Mass Communication, rev. ed., University of Illinois Press, 1971.

[Schulz von Thun 1981] F. Schulz von Thun: Miteinander reden 1 - Störungen und Klärungen. Psychologie der zwischenmenschlichen Kommunikation. Rowohlt, Reinbek, 1981.

[Shackel1991] B. Shackel: Usability - Context, Framework, Definition, Design and Evaluation. In B. Shackel & S. Richardson (Eds.): Human Factors for Informatics Usability (p. 21-37), University Press, Cambridge, UK, 1991.

[ShRP2007] H. Sharp, Y. Rogers, J. Preece: Interaction Design: Beyond Human-Computer Interaction. John Wiley & Sons, Hoboken, 2007.

[ShWe1971] C.E. Shannon & W. Weaver: The Mathematical Theory of Communication. University of Illinois Press, 1971.

[SiOp2005] G. Sindre, A. L. Opdahl: Eliciting security requirements with misuse cases. Requirements Engineering Journal, Vol. 10, No. 1, 2005.

[SmMa2011] S. Smith, R. Mazin: The HR Answer Book: An Indispensable Guide for Managers and Human Resources Professionals. 2nd edition, AMACOM, New York, 2011.

[Snijders et al.2015] R. Snijders, F. Dalpiaz, S. Brinkkemper, M. Hosseini, R. Ali, A. Özüm: REfine: A Gamified Platform for Participatory Requirements Engineering, Proc. 1st Int'l Workshop Crowd-Based Requirements Eng. (CrowdRE 15), p. 1–6, 2015.

[Snyder2003] C. Snyder: Paper Prototyping - The Fast and Easy Way to Design and Refine User Interfaces. Morgan Kaufmann, 2003.

[TiSi2017] S. Tiwari, S. Singh Rathore: A Methodology for the Selection of Requirement Elicitation Techniques. In: arXiv e-prints, 2017. https://arxiv.org/abs/1709.08481. Last visited September 2019.

[UXQB2017] UXQB Advanced Level - Usability Testing and Evaluation (CPUX-UT), Version 1.07, February 2017.

[Walten et al.2015] D.D. Walten, G.J. Roedler, K.J. Forsberg, R.D. Hamelin, T.N. Shortell: Systems Engineering Handbook – A Guide for System Life Cycle Process and Activitites. Wiley, 2015.

[Warfel2009] T. Warfel: Prototyping - A Practitioner's Guide. Rosenfeld Media, 2009.

[Wiki2017] Wikipedia Principle of abstraction: https://en.wikipedia.org/wiki/Principle_of_abstraction. Last visited February 2019.

[Withall2007] S. Withall: Software Requirements Patterns. Microsoft Press, 2007.

[YoAs2015] M. Yousuf, M. Asger: Comparison of Various Requirements Elicitation Techniques. In: International Journal of Computer Applications, Vol. 116, No. 4, 2015.