Grundlagen des Requirements-Engineerings

Bei der Entwicklung eines Systems – sei es nun eine Software oder ein Gebäude – ist die Vor-\n\aussetzung für den Erfolg, dass allen daran Beteiligten bekannt ist, was eigentlich entwickelt \nwerden soll. Das fängt bei den groben Zielen an und endet bei sehr detaillierten Vorgaben \nfür die Umsetzung.

Die Arbeit eines Requirements-Engineers befasst sich genau mit diesem Thema. Seine Auf-\nfgaben sind

• das Ermitteln,
• das Dokumentieren,
• das Prüfen und Abstimmen, sowie
• das Verwalten
von Anforderungen an ein System.

Wir möchten Ihnen im Rahmen dieser Broschüre einen Überblick darüber geben, welche \nTätigkeiten zu diesen Aufgabenbereichen gehören, und mit was wir, die SOPHISTen, uns seit \nmittlerweile fast 20 Jahren beschäftigen.

Die SOPHISTen

»Die kleine RE-Fibel«

The SOPHISTs

»A short RE Primer«
The SOPHISTs

SOPHIST GmbH
Vordere Cramergasse 13
90478 Nuremberg
Germany
www.sophist.de

Requirements Engineering

1st edition 2016

Translated from German by AMPLEXOR

Copy Editing & Production: Roland Kluge and Karin Schlör, SOPHIST GmbH
Cover Design and layout: Heike Baumgärtner and Alexander Holz,
SOPHIST GmbH
Printing and binding: Flyeralarm

Copyright SOPHIST GmbH

»A short RE Primer«

This work is protected by copyright. All rights, including translation, reprinting and reproduction of the book, or parts thereof, are reserved. This work may not be reproduced, edited or copied in whole or in part, by electronic means or otherwise (photocopy, microfilm or other media) without the written consent of the SOPHIST GmbH.
It should be noted that all software and hardware names as well as brand and product names of the respective companies are normally subject to the registered trademarks and patent protections.
While reasonable care has been exercised in the preparation of this brochure, the author and the SOPHIST GmbH assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.
Contents

1. Introduction and Motivation ................................................................. 7

2. What is Requirements Engineering?

2.1 A Requirement .................................................................................. 11

2.2 Differentiating Requirements .......................................................... 11

2.2.1 Levels of Detail ........................................................................... 11

2.2.2 Types of Requirements ................................................................. 13

2.3 Sources of Requirements .................................................................. 14

2.3.1 Stakeholders as Sources ............................................................... 15

2.3.2 Documents as Sources ................................................................. 18

2.3.3 Systems as Sources ....................................................................... 19

3. Elicitation of Requirements ................................................................. 21

3.1 Requirement Categories According to Kano ................................... 21

3.2 Creative Techniques .......................................................................... 23

3.2.1 Brainstorming ............................................................................. 23

3.2.2 Method 6-3-5 ............................................................................. 24

3.3 Observation Techniques ................................................................. 25

3.3.1 Field Observation ......................................................................... 25

3.3.2 Apprenticing ................................................................................. 26

3.4 Survey Techniques ............................................................................ 27

3.4.1 Interview ...................................................................................... 27

3.4.2 Questionnaire .............................................................................. 28

3.5 Document centric Techniques ......................................................... 29

3.5.1 System Archeology ....................................................................... 29

3.5.2 Reuse .......................................................................................... 29

3.6 Elicitation Techniques in Project Reality .......................................... 31

4. Documentation of Requirements ....................................................... 33

4.1 Mixed Documentation for Requirements .......................................... 34

4.2 The Requirements Template ........................................................... 35

4.3 Eliminate Linguistic Effects ............................................................... 36

4.3.1 Linguistic Effects .......................................................................... 36

4.3.2 The SOPHIST Set of REgulations ................................................. 38
1. Introduction and Motivation

We SOPHISTs have been working as consultants and trainers in requirements engineering (RE) for over 20 years. We support customers from many different industries, from pure software development, insurance and banking to the planning of building installations. The focus of our work is on the elicitation, documentation, validation and consolidation as well as the management of requirements.

The history of requirements engineering

The roots of requirements engineering (RE) date back to the 1970s. At that time, it required a great deal of effort to develop software systems. Since the needs of users were inadequately met in many cases, systematic RE began to emerge. Thus, the software industry was the cradle for RE, which has established itself in other sectors as well.

The first IEEE standard that defined RE more specifically followed in 1983. This became further elaborated on the basis of lessons learned and various experiences and the latest version, released in 2011, is ISO/IEC/IEEE 29148. The fact that the importance of RE is still underestimated in practice is explained in the following studies:

Studies conducted by the Standish Group in 2012 and published in the CHAOS Manifesto 2013 [Chaos 13] reveal that

- 74% of all projects did not meet the deadline,
- 59% of all projects exceeded the budget and
- on average, only 69% of the specified requirements were actually realized and implemented.

![Fig. 1: Time and budget overruns according to [Chaos 13]](image-url)
In particular, large projects with more than $10 million personnel costs often fail (38%).

A study by SwissQ [SwissQ13] shows that 45.1% of respondents view the maturity level of their RE as mediocre or weak. Also, in 65.4% of the cases there is “always” or “often” too little time dedicated to RE. This often results in inadequate requirements. 40.2% of respondents see the use of a defined RE process as a solution.

Requirements engineering has established itself in many sectors of the economy, as the relationship between RE and successful projects increasingly manifests itself. In order to convince you of the benefits of RE, this brochure gives you an insight into the work of a requirements engineer and shows you how to wisely elicit and document requirements - because good requirements are ultimately the basis for your project’s success.

A heartfelt thanks goes out especially to Chris Rupp for the strategic direction and planning, as well as to Susanne Appel and Pascal Geis for the content development and to Roland Kluge and Alexander Holz for design and layout but also to all the other SOPHISTs who have played their part in this brochure - the whole requirements engineering and management book team, the technical reviewers and the discussion partners who have contributed to the contents and the quality of this brochure.

2. **What is Requirements Engineering?**

A requirements engineer, also called systems analyst or requirements analyst, connects diverse areas in the context of a project. One of their main tasks is to maintain good contacts with the relevant stakeholders of a project.

**Stakeholders are all people or organizations that are affected by the development of a system as well as the usage and operation of the system or product. Examples of stakeholders are users of the system, operators of the system, developers, architects, customers and testers – hence, also people who are not involved in the development of the system, but use it later on, keep it in service or train the usage of the system.**

The definition of stakeholders according to [CPRE 15]:

**A stakeholder of a system is a person or an organization that has an (direct or indirect) influence on the requirements of the system.**

Their close contact with stakeholders allows requirements engineers to elicit and document what the stakeholders expect and demand from a system to be developed. However, since there is not always consensus among all stakeholders, a requirements engineer needs to mediate between the stakeholders with a lot of tact.

The work of a requirements engineer involves a systematic framework known as requirements engineering.

**Requirements engineering describes a systematic approach, from the project idea and the objectives to a full set of requirements.**

Yet the work of a requirements engineer does not end with the elicitation and documentation of the requirements. To ensure the quality of the requirements, they must be examined on the basis of predetermined criteria. In addition, the requirements must be agreed on with the stakeholders in order to identify conflict areas and to resolve such conflicts.

Last but not least, it is the responsibility of the requirements engineer to manage the agreed requirements. Indeed, in order to be able to precisely understand changes in the...
What is Requirements Engineering?

requirements or in the requirements documents, one must be able to track the life cycle of a requirement. As such, requirements management forms an essential part of requirements engineering.

Fig. 2: The main responsibilities of a requirements engineer

In the following section we address the different types of requirements, the possible sources of requirements and how to elicit requirements from the available sources.

2.1 A Requirement

Before we discuss the various types of requirements, we should first ask ourselves what requirements actually are.

We SOPHISTs have found what we believe to be a suitable definition for a requirement, which in practice has proven to be understandable, comprehensive and sufficiently precise:

**Definition of a requirement according to SOPHIST**

*A requirement is a statement concerning a property or the performance of a product, a process or the people involved in the process.*

A requirement is therefore more than the performance to be expected of a system or product. Requirements can also refer to test cases or manuals, determine a defined procedure or concern the qualifications of system engineers. All of these are examples of requirements that must be met within the framework of a project.

2.2 Differentiating Requirements

Requirements describe various aspects of a system, of a procedure or of persons and therefore come in very different forms. We describe three ways to distinguish requirements below: the level of detail, the category and the type of requirement.

2.2.1 Levels of Detail - Begin with the End in Mind

In order to describe a system to be created, it is advisable to first get an overview of the entire system, for example by clarifying the objectives of the system, its main features and how it is to be embedded in the business context. An abstract description is sufficient to set the vision and it forms the uppermost level of detail of a system (specification level 0). Below this top level of detail are four further specification levels.

By refining the rough requirements and adding further information during the system analysis, the lower specification levels are filled with requirements until eventually enough information is available in the form of requirements to develop the system.
### 2.2.2 Types of Requirements

Requirements can also be grouped according to their nature. Based on our experiences in the field, a classification into the following requirement types has withstood the test of time:

<table>
<thead>
<tr>
<th>Requirement Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional</td>
<td>User interface requirements, Quality requirements, Functional requirements, Legal / contractual requirements, Activities and services, Requirements for other deliverables</td>
</tr>
<tr>
<td>Non-functional</td>
<td>Technological constraints, Requirements that describe the system functionalities and features in detail and general, functional agreements and restrictions</td>
</tr>
</tbody>
</table>

In this subdivision, the rough subdivision in functional requirements and non-functional requirements is reflected.

As such, functional requirements describe:

- actions that are carried out autonomously by a system,
- interactions between the system and a user or another system,
- or general, functional agreements and restrictions.

Every requirement that is no functional requirement, is non-functional. The non-functional requirements can be roughly grouped into two categories:

- Requirements that describe the system functionalities and features in detail and
general, functional agreements and restrictions.

Technological requirements, quality requirements and user interface requirements usually relate to a function within the system.

Non-functional requirements for other deliverables, requirements for activities and services and legal and contractual requirements go more across systems or are project-related.

Detailed information on the types of requirements can be found in [Rupp14].

---

### Fig. 4: Types of requirements

- = functional requirements
- = non-functional requirements

---

### Fig. 3: Levels of detail

The specification levels described here are relevant for different stakeholders. For management, a glance at the top-level specification is usually enough to grasp the overall scope and objectives of the system, as well as the supporting business processes.

At the middle specification levels (2-3), the technical (sub)processes are defined. In addition, tasks can be assigned to the participants in the process. The middle specification levels therefore are the most interesting for the functional departments.

Architects and developers also deal with the lowest specification level, level 4, which separates the system in question into software and hardware or other respective components and where technical details can be found.

For a full description of the system, all specification levels should be specified in such a way that sufficient information is available to all project members. That does not mean, however, that every single requirement needs to be refined all the way down to specification level 4.

---

### Table: Levels of detail

<table>
<thead>
<tr>
<th>Specification level</th>
<th>Assigned terminologies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0</strong> Rough system description, system objectives, system overview, vision, introduction, mission statement</td>
<td>Scrum, RUP or V-Model</td>
</tr>
<tr>
<td><strong>1</strong> Use case, user story (application) scenario, technical conception, functional description, functional outline, technical requirement, organizational requirement, list of features, context boundaries</td>
<td>Epic, Theme</td>
</tr>
<tr>
<td><strong>2</strong> User requirement, operational concept description, operational requirements description, interface requirements specification, specification sheet, target concept, rough specification, operational requirement, test cases, list of features</td>
<td>User Story, Technical User Story, Backlog Item, Acceptance Criteria, Definition of Done</td>
</tr>
<tr>
<td><strong>3</strong> Detailed user requirements, technical requirements, interface overview, interface description, system segment specification, interface requirements specification, system requirements, detailed specifications, test cases, list of features</td>
<td>Technical User Story, Backlog Item, Acceptance Criteria, Definition of Done</td>
</tr>
<tr>
<td><strong>4</strong> Component requirements, technical requirements, interface overview, interface description, software requirement specification, interface design description, requirements specification, detailed specifications, module requirements, test cases</td>
<td>Technical User Story, Backlog Item, Acceptance Criteria</td>
</tr>
</tbody>
</table>
What is Requirements Engineering?

Why do you need non-functional requirements?

The elicitation and documentation of non-functional requirements takes a lot of effort. Therefore, they are, unfortunately, often neglected in practice, although they are essential for the project’s success.

Only when you elicit the non-functional requirements in addition to the functional requirements do you get

• a complete requirements specification
  A complete requirements specification also includes non-functional requirements. Architects and developers rely on this information to properly implement the system, especially when it comes to the required performance of the system.

• satisfied customers
  In particular, good quality requirements and requirements for the user interface turn customers into satisfied customers.

• reliable planning and legal certainty
  Requirements for various deliverables, for actions to be carried out, as well as legal and contractual requirements determine what is expected of which party throughout the course of a project. This allows potential litigation to be absorbed beforehand.

Non-functional requirements help all stakeholders to gain more clarity regarding the system and the development process.

2.3 Sources of Requirements

Requirements describe all functional and non-functional aspects of a system and thus form the basis for the success of a project or system. The question is, however, how to arrive at the requirements.

There are basically three different sources of requirements:

• Stakeholders: Persons or institutions who directly or indirectly impact the system.
• Documents: Laws, standards, manuals or other documentation may be used to determine requirements.
• Systems: It is often helpful to analyze a previous system or a competing product.

2.3.1 Stakeholders as Sources

Our experience has shown that stakeholders possess a lot of knowledge - customers often have visions of what a system should be able to do. Users can provide information regarding system operation. Architects and developers know what is feasible. This knowledge is often anchored in many different minds, and the challenge of the requirements engineer is to gather all the information and arrive at a consensus so that in the end all the necessary requirements have been documented and are accepted by all parties.

The first step is to find out who the relevant stakeholders are since it is not advisable to take too many stakeholders on board from the point of view of time and costs alone. Each group of relevant stakeholders, for example users, buyers, testers, system architects, etc. should be represented by a maximum of two people. Which person is best suited from each group depends on many factors. The first point to clarify is what kind of knowledge each stakeholder possesses and who is available. Such information can only be obtained through good communication.

In order not to lose any of the stakeholders’ knowledge and to share it with other people, it is advisable to write the stakeholders down in a list. It should be saved so that all authorized persons can access it at any time.

Which information belongs in a stakeholder list?

Only important information should be found in a stakeholder list because it quickly becomes confusing otherwise. Which information is useful and important for a project can vary. From our experience, however, we can report that the following attributes have proven to be the backbone of a stakeholder list:

• Stakeholder’s name
• Function in the company or role in the project
• Stakeholder’s contact details
• Availability: When and how can the stakeholder be reached
• Knowledge: Field and scope
• Relevance of the stakeholder
• Stakeholder’s interest in and objectives for the system

These attributes are all depicted in the stakeholder list.
What is Requirements Engineering?

In order to capture the relevant information on key stakeholders quickly, a stakeholder list should be as short as possible and as comprehensive as necessary. If you manage to include a representative of each stakeholder group in your list in the requirements analysis, this will be a great help to you during the project, when it comes to determining the requirements as completely as possible. If none of the stakeholders of a group can be nabbed, a persona serves as a substitute.

What are personas?

Personas are fictive characters that are based on data from market research, surveys and lessons learned and describe the various objectives and observed behavioral patterns of potential stakeholders. Thus, they can represent a group of stakeholders, such as the users of a system.

From experience and market research data, archetypal users are first extracted based on age or gender and are then further described in more details.

To design a realistic persona, along with attributes such as name, function / role, knowledge and goals, personal characteristics are also helpful. A photo of the persona and their age, hobbies, likes and dislikes, family status and residence breathe life into them.

<table>
<thead>
<tr>
<th>Role</th>
<th>Cook</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Tommaso Zanolla</td>
</tr>
<tr>
<td>Age</td>
<td>34</td>
</tr>
<tr>
<td>Family status</td>
<td>married, 2 children</td>
</tr>
<tr>
<td>Salary</td>
<td>low</td>
</tr>
<tr>
<td>Residence</td>
<td>outside of the city</td>
</tr>
<tr>
<td>Origin</td>
<td>Italy</td>
</tr>
</tbody>
</table>

**Activities**

- **Managing procurement and inventory**
  Is responsible for ensuring that enough food and drinks are available during the week. In addition, he is also responsible for the inventory.
- **Prepare dish**
  The cook prepares the dish ordered.
- **Place orders**
  Orders that do not involve food are entered onto a prescribed form.

**Goals**

- **Centralized procurement**
  Wants to access the "Suppliers Table" at any time.
- **Display stocks**
  Wants the system to show the inventory.
- **Comparison shopping**
  The cook wants to have a price comparison of the products when shopping.

**Incentives**

- **Online shopping**
  Since he has little time to go shopping, he prefers online shopping.
- **His children**
  In his free time, he enjoys being with his children the most.

**Fig. 5: Stakeholder list**

**Fig. 6: Example persona**

A persona helps to determine the requirements of absent stakeholders by considering the perspective of the persona as representative for a group of stakeholders, when collecting data regarding the requirements and documentation of their opinion.

In addition to stakeholders and personas, documents are also an important source of requirements.
2.3.2 Documents as Sources

Documents often contain relevant information, from which requirements can be deduced. Universally valid documents such as norms, standards or legal texts can tremendously influence the course of the project, for example when specific development standards are demanded. Moreover, there are industry or organization specific documents, such as bug reports or training documentation from the predecessor system, specifications or documentation for business processes that can serve as a source.

Once again, all relevant documents should be managed in a list. Helpful attributes for a document list are:

- Document name
- Document status
- Document type
- Contact person (if available)
- Storage location
- Content
- Level of confidentiality

A document list can look like the following:

<table>
<thead>
<tr>
<th>Document 1</th>
<th>Document 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Online help for university catalog COPAC</td>
</tr>
<tr>
<td><strong>Document status</strong></td>
<td>Release</td>
</tr>
<tr>
<td><strong>Document type</strong></td>
<td>Online help</td>
</tr>
<tr>
<td><strong>Contact</strong></td>
<td>TU Denkhausen</td>
</tr>
<tr>
<td><strong>Storage location</strong></td>
<td><a href="http://www.tud-bib-support.de">www.tud-bib-support.de</a></td>
</tr>
<tr>
<td><strong>Content</strong></td>
<td>Search function for borrowing, extension for objects on loan, creating bibliographies/checklists</td>
</tr>
</tbody>
</table>

Fig. 7: Example of a document list

The attributes should be determined completely and, if necessary, supplemented. This makes work easier for everybody involved. Existing contacts for documents should be kept up-to-date in the stakeholder list.

2.3.3 Systems as Sources

Systems in operation can be predecessor systems, as well as competitive or neighboring systems.

Lessons learned with these systems help to determine requirements - be it when you expand or change a system, or a new development.

- You want to understand the processes of a system, but have no documentation on hand? The analysis of a (predecessor) system will help you gain a better understanding.
- A system should be redeveloped? Have a look at your competitors’ systems and decide for yourself which functions and non-functional aspects are relevant for you.
- You have interfaces to neighboring systems? Analyze the interface systems in order to obtain an insight into what data you need to provide and what you receive.
- You need support systems to build your system? Examine potential support systems in order to build the ideal system for you.

Manage all relevant systems in a system list, in which the important attributes are given, in order to determine and manage important aspects at a glance.

In doing so, the following system characteristics should, if possible, be recorded:

- Name
- System type
- Development status
- Source
- Usage
- Content

This results in a system list similar to the following one:
### 3. Elicitation of Requirements

Having identified all the relevant information sources, documented the information in lists and stored it so that it is accessible to all relevant persons, the sources for eliciting requirements are now utilized. Elicitation techniques are used to reach the relevant information.

Elicitation techniques can be divided into four categories:

- Creativity techniques
- Observation techniques
- Survey techniques
- Document-centric techniques

Moreover, support techniques (as e.g. mindmapping, SOPHIST Set of REgulations or Scenarios) will be of use.

Whatever elicitation techniques you choose depends, among other things, on the requirement category. We use the findings from research into customer satisfaction conducted by Dr. Noriaki Kano when classifying requirements.

#### 3.1 Requirement Categories According to Kano - Stakeholder Satisfaction

For eliciting requirements it is essential to know what impact requirements have on the stakeholder satisfaction. Dr. Noriaki Kano introduced the so-called Kano Model, which divides the features of a product in the following three categories, as early as 1978:

- **Basic factors** are naturally presupposed features. If basic factors are not implemented in a product, this leads to massive dissatisfaction on the part of the customer - the success of the product is endangered. Basic factors are taken for granted by a stakeholder and therefore the stakeholder will not address them explicitly. Basic factors are attributable to subconscious knowledge. Basic factors are determined primarily using observation and document-centric techniques, by which you can get information that was not “pre-filtered” by stakeholders and sorted out as being obvious.

- **Performance factors** are consciously expected system features. The more performance factors are present, the more satisfied the customer is with the product. Performance factors are quite present in the stakeholder’s consciousness (conscious knowledge). Therefore, in surveys regarding system requirements performance factors are often mentioned first. Survey techniques are therefore your first choice when investigating.
Excitement factors are the features of a product that the customer does not know about beforehand and discovers only during use as a pleasant surprise. Stakeholders are not aware of them (unconscious knowledge), yet they lead to above-average enthusiasm on the part of the customer. Excitement factors are determined primarily using creativity techniques, as you best can create an atmosphere that favors new ideas and innovations in this context.

Over time, excitement factors turn into performance factors and eventually into basic factors.

When excitement factors catch on in the marketplace, more and more producers start to offer similar features. Customers begin to get used to the features and, over the course of time, excitement factors turn into performance factors and ultimately basic factors. This means that, in order to remain a market leader, you need to always remain creative.

3.2 Creativity Techniques

In order to develop new, innovative ideas and the first visions of a system, you need to break away from conventional thought processes in order to let the creative juices flow and to also link unfamiliar concepts. An important aspect of this is to establish a suitable environment for creativity. Only in this way can sinking in a creative chaos be avoided.

3.2.1 Brainstorming

A widespread technique for encouraging creativity in a group of stakeholders is brainstorming. With this elicitation technique, a group of 5-10 people gathers together ideas in a fixed time frame. Every idea is first written down without review by a moderator, no matter how crazy or unusual it may seem. Since all participants in a brainstorming session are exposed to all the ideas, they can use the ideas of the other participants to develop their own new ideas. After the prescribed time has elapsed, the collected ideas are subjected to a detailed analysis.

Brainstorming is most effective when different stakeholder groups participate and good group dynamics are present. When participants encourage each other to find new ideas, brainstorming can be a lot of fun and often leads to good results.

If the group feels inhibited, however, creativity is blocked. The moderator definitely must make sure therefore that the collection phase is free from criticism and negative comments because that is how good ideas get lost.

Brainstorming paradox is a variation of brainstorming. With this variation, the idea is to gather ideas on objectives that are NOT to be reached. Examples are: “What do we need to do so that the project fails?” or “How do we ensure that our product is a flop?” The procedure for generating ideas is the same. The main difference becomes clear when we analyze the results. In the case of brainstorming paradox, measures are primarily developed on how the results obtained can be prevented.

In order to include other perspectives when brainstorming, it may be worthwhile, in addition to the professional stakeholders, to involve other people. For example, a working student, the facility manager or a trainee can provide new input since they are less bogged down in prevailing thought patterns and processes.
### 3.2.2 Method 6-3-5

The 6-3-5 method is a creative technique in written form, by which six participants each note down three ideas on a piece of paper and pass on this note to their neighbor after a fixed time period. The new piece of paper with the neighbor’s three ideas is a source of inspiration and thus stimulates the participants to find new ideas, which in turn are listed. This is repeated until each participant has had each list once, in effect five times. Following this, the ideas from all the pieces of paper are collected and evaluated jointly.

The 6-3-5 method has some advantages, especially compared to brainstorming. As a written exchange it can be readily used to avoid emerging conflicts even in the case of difficult group dynamics and distributed teams. Moreover 6-3-5 is easier to cope with, especially for less experienced moderators, because, unlike brainstorming, a lot of ideas are not being thrown around the room at the same time that need to be written down by the moderator.

The aim of the method is primarily to develop existing ideas, but new ideas should not be categorically ruled out.

The disadvantage of the 6-3-5 method is that collective creativity in writing cannot be as lively as direct, verbal communication. Particularly for low-key participants this can, however, be very productive.

### 3.3 Observation Techniques

Our practical experience has shown that many stakeholders are indeed knowledge providers, yet they have difficulty explaining this knowledge. Moreover, it is sometimes a big problem that the stakeholders do not have enough available time. If you are confronted with such problems - and that is probably the case in every project - we recommend the use of observation techniques, in particular for determining basic factors.

Observation techniques involve the requirements engineer observing a stakeholder, usually a system user, while he works. The work steps are written down and an analysis is made of which work processes are relevant to the stakeholder. The stakeholder can also play an active role in order to impart his knowledge or demonstrate something to the requirements engineer.

Observation techniques work only if there is a system in which work processes / application processes can be well observed. However, the requirements engineer should, in this case, be aware that in doing so the actual situation is analyzed, which may result in documentation of obsolete technologies or processes in need of improvement.

#### 3.3.1 Field Observation

When it comes to field observation, the requirements engineer captures the activities of the stakeholder, including their temporal dependencies, as well as often unconsciously carried out work steps, which belong to complex processes. From the perspective of the requirements engineer, it is useful to familiarize himself with the working environment of the stakeholder in advance. To capture all relevant aspects and yet be able to understand everything in retrospect, the use of video recordings has proven itself valuable. However, this and the reuse of the recordings must be discussed with the stakeholder and the project manager.

The stakeholder carries out his work steps as usual and the requirements engineer observes them without the work processes being interrupted with questions and explanations. This allows processes and deviations from given processes to be effectively uncovered. The information gathered usually corresponds with basic factors, but in some cases with performance factors.
3.3.2 Apprenticing

Apprenticing denotes the learning of new stakeholder activities under supervision. This helps the requirements engineer to get an accurate picture of the work processes - especially if these are hard to observe.

Using the findings, the requirements engineer is able to derive detailed requirements for support systems, as well as test cases and potential error cases. However, only the actual process is analyzed, requirements for the target process are left out.

Apprenticing is suitable when processes that are difficult to observe should be followed. These often require an explanation from an experienced stakeholder. Giving another person instructions is usually an effective technique for stakeholders who otherwise cannot put their knowledge into words very well. Afterwards, by working alone on the system the requirements engineer collects valuable experience and information.

However, that is exactly the catch to this elicitation technique due to the fact that, especially in critical systems such as air traffic control, operation by the requirements engineer can cause danger, making it therefore impracticable. In addition, it costs the requirements engineer, in particular, but also the apprenticing stakeholder a considerable amount of time and expense, and is therefore only profitable for longer-term projects.

3.4 Survey Techniques

The most commonly used elicitation technique is survey techniques, during which stakeholders are specifically asked about their wishes and needs. Performance factors, in particular, can be determined since these are usually very present in the minds of the stakeholders (conscious knowledge). The success of survey techniques is highly dependent on the linguistic skills of the participants, especially non-functional aspects are difficult to put into words. ‘Survey techniques’ includes the performing of both oral and written methods.

3.4.1 Interview

The classic form of elicitation techniques is the interview. The requirements engineer asks one or more stakeholders questions and records their answers. Interviews exist in various forms:

- **Standardized interview:**
  - The order of the questions is determined in advance.
  - The exact wording of each question must be kept.
  - All possible answers are considered in advance.

- **Partly-standardized interview:**
  - The interview consists of a predetermined and an open part.
  - The open part of the interview allows for free questions and answers.

- **Non-standardized interview:**
  - Only the topic or a rough guideline is set down.
  - Existing sets of questions can be used.

The difference therefore is based on the degrees of freedom with which the interview is conducted. Depending on the focus, one then consciously chooses one of the interview types mentioned. A standardized interview is useful for collecting data from multiple stakeholders on a specific subject in order to obtain a statistical analysis, while a non-standardized interview allows for individual conversation.

This may, on the one hand, lead to very good results, especially when the speaking skills of the requirements engineer are well developed. On the other hand, it carries an inherent risk that you may deviate too much from the real issue at hand and wind up at the end without any result, i.e. requirements.
A good interview is characterized by the fact that it is well-prepared. Both parties should have dealt with the topic prior to the interview. The interviewer must therefore inform the stakeholders in advance about the topic of the interview and clarify the location and the duration of the interview.

The questions in the open part of an interview should be formulated to be as neutral as possible - leading questions falsify the results of the interview and may cause the stakeholder to reveal less, or even no information at all if he feels manipulated.

A well-managed interview can bring excellent results due to the fact that secondary aspects can be taken up and specific points can be responded to immediately. However, since an interview takes a lot of time, including the preparation and review, selecting the right interview partner is crucial. Draw on your stakeholder list and pick out the representatives who can provide you with the maximum amount of relevant information for each stakeholder role.

3.4.2 Questionnaire

A questionnaire provides a way to survey stakeholders on paper. Depending on the objective, a questionnaire may contain open and closed questions. Open questions provide the stakeholders more opportunities to answer and are therefore sure to uncover opportunities for improvement, while closed questions are automatically evaluated and are therefore particularly suitable for surveying numerous stakeholders.

The success of using questionnaires depends heavily on whether and how seriously the surveyed stakeholders fill out the questionnaire. You should therefore ensure that all stakeholders know the benefits to be obtained from the answers and are motivated to answer the questions truthfully. Experience also shows that important questions should be asked at the beginning of the questionnaire, otherwise you run the risk that they are left unanswered if someone does not fill out the questionnaire completely.

3.5 Document-centric Techniques

The desire to further develop a system may lead to the realization that domain-specific business logic was used in the old system, but that it is largely unknown to the staff. There are various reasons for this. Perhaps the system was not developed in-house. Employees who were involved in the development have long since left the company, etc. This means that sometimes only the system itself and its documentation are left as requirement sources.

With document-centric techniques, you can determine very well how the functionalities of a system are implemented. Thus, you get information about already implemented performance factors and even basic factors that have been completely internalized by stakeholders and are therefore no longer spoken of in interviews, for example. Before using document-centric techniques, determine which parts of the old system should be transferred to the new system and which parts are to be redeveloped.

3.5.1 System Archeology

When it comes to the system archeology, you use the existing system itself or the associated documentation, e.g. user manuals, online tutorials, strategy papers or possibly even code, in order to get an overview of the behavior of the system and filter out requirements using extraction techniques [John03]. Keep it in the back of your mind that these requirements are based on the actual situation and assess whether these requirements are still valid.

Start with documents from which the features of the system are as easy as possible to see, such as the user manual or old test cases. This approach also enables you to gain a first overview of the professional domain. Should questions still remain unanswered after analyzing these “introductory documents” or you need more details, you still have the opportunity to analyze other documentation on lower specification levels all the way down to the code.

3.5.2 Reuse

If a similar system has already been developed, often requirements and other artifacts can be reused from the previous project. Check which artifacts – especially requirement documents – are available from the previous project. Especially repetitive processes, such as “Search” or “Save” are ideal candidates for reuse.

With a little preparation, the effort required for reuse, especially to find relevant requirements of sufficient quality, is greatly reduced. For example, on an appropriate level of abstraction (like use cases or test cases) you can generalize and put requirements into an experience database. That way, the requirements only need to be adapted to the project specifics in
the next project. Although this causes extra work initially, going to the effort of determining and describing already known functionalities can be omitted in the follow-up projects in the best case scenario.

In addition to the previously mentioned elicitation techniques, there exist a number of other techniques that cannot be explained at this point for sheer lack of space. More information can be found in [Rupp14].

3.6 Elicitation Techniques in Project Reality

The theory of elicitation techniques presented so far is easy to learn. Selecting the appropriate elicitation method, however, is not always easy, since it is influenced by many factors. These factors have been summarized in the matrix on page 32 (see Fig. 10).

The matrix is based on our experiences from project activities and numerous discussions. Depending on the project itself, your evaluation criteria may change or other factors can play an important role. Therefore it is especially important to understand the basic idea behind the matrix.

How is the selection matrix used for elicitation techniques?

1. Decide in advance what kind of knowledge (unconscious, subconscious or conscious knowledge) you want to collect in order to reduce the choices already in the first step. By doing so, you determine whether creativity, observation, survey or document-centric techniques should be applied.

2. Analyze the influencing factors that are present. Highlight the three or four key influencing factors in the table which are most pronounced in your project.

3. Now look for the elicitation techniques from the group formed in step 1 that have the highest rating for you with respect to the factors from step 2.

In order to fully collect all requirements, basic, performance and excitement factors need to be determined. This usually happens only by employing a mix of elicitation techniques. Moreover, we recommend the use of support techniques such as video or audio recordings, prototyping and reducing processes to their basic terms in order to safely reach your target.
4. Documentation of Requirements

A lot of information can be collected through the successful use of elicitation techniques. The documentation of this information not only serves to make this knowledge available to all project participants, but it also helps to structure the accumulated knowledge. When it comes to the type of documentation, you have the choice between natural language and requirement models, or a mixture of both.

Natural language vs. model-based requirements

Both natural language and model-based documentation have advantages and disadvantages. Natural language documentation is characterized by the fact that it is not necessary to learn special notation because natural language is understandable for everybody. Moreover, natural language is suitable for the documentation of any kind of requirements. However, caution is required because natural language is often ambiguous or misleading.

Model-based requirements documentation, however, lends itself very well to look at the system from different perspectives one-by-one and isolated, for instance the purely structural view of the data to be processed, the functional view of workflows or the behavioral perspective, which illuminates, among other things, system responses to events.
Thanks to the compact and, for the experienced reader, clearly understandable depiction, misunderstandings can be avoided. The disadvantages, however, are obvious - the corresponding notation has to be learned by everyone involved. In order to present complex issues using models, a single chart type is usually not sufficient. Models are therefore not universally applicable.

A common approach in practice is to combine model-based and natural language requirements in order to take advantage of both forms.

### 4.1 Mixed Documentation for Requirements - the STABLE Approach

The STABLE technique developed by SOPHIST GmbH (STructure and Administrate Requirements Better, Legibly and Efficiently) is a tried-and-tested approach, which supports the systematic structuring of functional requirements. The starting point here is a use case analysis, a type of system analysis by which the system under consideration is broken down into behavioral elements from coarse to fine.

It begins with a collection of use cases that abstractly describe the system functionality from the user perspective. Through further analysis these become more detailed. Here finer use cases, activities or states are helpful. Often, the results of this analysis will be documented in the form of UML diagrams. For further information on UML diagrams and use case analysis see [Rupp12].

The results are then collected and put almost 1:1 into an outline, which can then form the chapter structure for natural language requirements.

The system <system> [<condition>] SHALL PROVE <actor> WITH THE ABILITY TO <process verb> <object>

Fig. 12: Transfer from use cases into the outline - the creation of the basic structure

[Rupp14] gives you assistance by describing the necessary steps from the use case analysis towards an outline structure.

STABLE helps you to create a simple structure for your functional requirements. In this structure, you can also classify non-functional requirements, provided they are assignable to a use case, an action or a state. The resulting outline structure thus forms the basis of your requirements specification for natural language as well as model-based requirements. Below we will look a bit more closely at the documentation of requirements in natural language.

### 4.2 The Requirements Template - A Tool for All Cases

The SOPHIST requirements template is a blueprint that defines the structure of a single requirement sentence. The structure of the individual requirements is hence unified and you can already tell at first glance whether or not important “components” are missing. Particularly when specifying in a foreign language, a prescribed requirements framework can make it easier to overcome uncertainties.

Using the requirements template is easy to learn and reduces unwanted linguistic effects since the syntax for writing a requirement is already provided. Compared to arbitrarily formulated prose requirements, the quality of the requirements is already visible after the first application. The SOPHIST requirements template for functional requirements has become an integral part of most requirements engineering processes in companies and corresponds to the FunctionalMASTER.

Based on our practical experiences, we have further developed the concept in order to cover, in addition to the functional requirements, also non-functional requirements and conditions with our template. For further details about the FunctionalMASTE R, its application for...
specifying in English, and our new templates, see our brochure “MASTeR - Schablonen für alle Fälle” [SOPHIST13] and Requirements-Engineering und Management [Rupp14].

4.3 Eliminate Linguistic Effects - The SOPHIST Set of REGulations

The use of a requirements template in combination with the SOPHIST Set of REGulations will help you to reduce linguistic defects when formulating your requirements. But what are linguistic defects and where do they come from?

4.3.1 Linguistic Effects

Everyone perceives their environment differently - psychologists claim that the subjective reality of the individual differs from the existing reality. The personal knowledge of the individual is reflected in the totality of their perceptions. This is influenced by social characteristics, knowledge and lessons learned.

Once people communicate their knowledge, transformation processes take place that are dependent on how the speaker assesses the situation and how his counterparts, which knowledge they, for instance, presuppose the other to possess or how confident they feel about their knowledge.

Knowledge passes thus from its creation until its absorption by the requirements engineer through different phases of transformation:

- **Perceptual transformation:** Every individual perceives reality differently.
- **Representational transformation:** A modification occurs when knowledge is put into language.

These transformation processes may result in a loss or a corruption of the information.

Perceptual transformations cannot be resolved - if information was not perceived by a stakeholder, even the best requirements engineer will not be able to conjure it up - but the impact can be compensated for by interviewing more than one person on the same matter, so as to obtain an overall picture.

Representational transformations, however, can be very easily resolved - the precondition is that the requirements engineer knows the possible transformation types and their consequences. This is where the SOPHIST Set of REGulations comes in. It is essentially based on the meta model of language used in neuro-linguistic programming (NLP) [Bandler75] [Bandler94].
Each of the transformation categories – deletion, generalization and distortion – is reflected in specific linguistic effects. Effects can lead to requirements of inferior quality. However, not all effects result in actual defects.

You should therefore decide for yourself, if a linguistic effect poses a problem and should therefore be remedied – then, in your perspective, the effect is an actual defect. The distinction depends on certain determining factors. A major point is the level of detail of the requirements you are currently writing.

Linguistic effects are quite normal and not always avoidable when writing generic requirements (Specification level 0 and 1). When writing more detailed requirements (Specification level 2 - 4), linguistic effects are more harmful and should always be critically examined. It is your job to detect these linguistic effects and remove the defects if the missing or distorted information is crucial. The SOPHIST Set of Regulations will help you to identify and correct the linguistic effects.

Please see [Rupp14] for more information on the distinction between effect / defect.

4.3.2 The SOPHIST Set of Regulations

Fortunately, people are governed by rules when using natural language to formulate knowledge. Thus, there are signs in natural language statements, whether oral or in writing, if one or more of the above transformation processes have taken place. The SOPHIST Set of Regulations is based on these unconsciously applied rules and makes it possible during the system analysis to locate ambiguous and contradictory statements in requirements documents in a defined and systematic way.

The SOPHIST Set of Regulations is a technique that allows you to recognize deletions, generalizations and distortions in requirements, and to thus uncover missing and distorted information.

How do I use the SOPHIST Set of Regulations?

The SOPHIST Set of Regulations provides you with a methodological toolbox, containing a total of 18 rules for the systematic analysis of your requirements.

Analytical application of the Set of Regulations on the basis of generated requirements.

For each rule in the SOPHIST Set of Regulations, the following three steps are performed:

1. Identifying linguistic effects in requirements based on signal words.
2. Analyzing lost or corrupted information through targeted questions.
3. Clearing away linguistic deficiencies or material errors by reshaping the requirement using the answers given so that the effects are eliminated.

If you follow this analytical agenda, you can improve your requirements step by step.

When starting to work with the SOPHIST Set of Regulations, it is advisable to use only part of the 18 rules. Based on our experience, we have assigned priorities to the rules to help you to first identify those linguistic defects which lead to the most serious errors in the later course of the project:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Resolve nominalizations that are not exactly defined and write one or several new requirements with a “good” main verb for every nominalization.</td>
</tr>
<tr>
<td>6</td>
<td>Ask wh-questions about the main verb.</td>
</tr>
<tr>
<td>12</td>
<td>Question vague nouns.</td>
</tr>
<tr>
<td>17</td>
<td>Requirements with incomplete conditional structures should be checked and formulated or described by another requirement.</td>
</tr>
<tr>
<td>18</td>
<td>Write one or more requirements for every implicit assumption not described.</td>
</tr>
</tbody>
</table>

Fig. 15: SOPHIST Set of Regulations rules with the highest priority

With experience, you can then gradually add other rules. Also, use the requirements you write down first as a “test basis”. The more you have internalized the rules, the easier it will be for you, even in conversational situations such as interviews, to pay attention to linguistic effects and already use the rules when formulating natural language requirements.

Constructive usage of the SOPHIST Set of Regulations

The SOPHIST Set of Regulations can be roughly divided into the following three processes:

- Checking the individual words (components) of a requirement statement (rule 1-6)
- Checking the requirement statement as a whole (rule 7-15)
- Checking the requirement statement in the overall context (rule 16-18)
The most important rules in the SOPHIST Set of Regulations

Rule 3: Resolve nominalizations that are not precisely defined.

Nominalizations are nouns that capture complex processes in one word ("verbal noun"). This conversion saves you the hassle of having to describe processes in detail. The danger is that, in doing so, the processes themselves become obscured and information that is essential for the description of the process will be lost by the nominalization.

Therefore, analyze each nominalization and check whether the process is sufficiently specified. If this is not the case, for each nominalization
- specify one or more new requirements, each with a "good" main verb, OR
- create a new glossary entry.

The library system shall facilitate borrowing.

Behind the nominalization 'borrowing', a complex process combining a large amount of information is hidden. Hence from the requirement it is not sufficiently clear
- who is authorized to borrow media (i.e. Is a library card needed? Are there age restrictions to pay attention to?)
- which items can be borrowed
- are there any costs
- which actions are performed by the librarian during the borrowing process.

The nominalization in the requirement leaves a lot of questions unanswered. This often indicates a distinct use case of its own in your requirements document. If you are able to answer any of the questions above in your requirement document, you can leave the above requirement as it is. Otherwise, you will find that resolving the nominalization results in several additional requirements.

Rule 6: Analyze missing information regarding the main verb.

In order to clearly describe a process in a requirement, it is necessary that all the information needed for the complete explanation of the main verb (process word) is contained in the requirement. If questions about the process are still open, the corresponding information must be detected and added to the requirement.

Scrutinize the main verb using the typical wh-questions. If you cannot answer all the questions with the given information, some information was deleted. If the missing information is relevant, then fill in the missing information.

The library system shall provide the user with the ability to display information about items that can be borrowed.

From this requirement it is not clear what, when and how information is displayed. Therefore, ask the key questions surrounding the main verb. By answering the questions "when", "how", "where", and "how often", it can be established whether the process word is sufficiently specified or whether the requirement must be supplemented with more information.

Eliminating the inadequately specified process words can lead to time-consuming reformulations of the requirement. Often, additional requirements result from the questions surrounding the main verb, but you might also recognize that the original requirement must be refined by adding more detailed requirements and thus replaced.

Rule 12: Scrutinize vague nouns.

In a requirement, both actors and objects are represented by nouns, for which a type of behavior or a property is required.

If a noun does not describe a clearly grouped set of objects, such information was possibly erroneously summarized. You should therefore get to the bottom of ambiguously formulated nouns and check whether the desired objects or actors have actually been designated by the nouns.

The system shall be able to display statistics graphically.

In this requirement, the noun "statistics" needs to be observed more closely. Do "statistics" refer to statistically calculated data from the items borrowed, figures relating to the library system and how often it was accessed, or customer data? Clarify who or what is being referred to by vague nouns and which part of the said amount the noun aptly describes.
Rule 17: Analyze incomplete conditional structures.

Each conditional situation requires a description of what should happen if the condition does not occur. For this case (or these cases), clarify the system behavior, as well as each condition that was not yet described.

If an item is not reserved, the library system shall provide the librarian with the ability to continue with the borrowing process.

The requirement leaves the question open of how the system should act in the case of a reserved item. If the respective system behavior has not been described, clarifying the question of the system response leads to an extension. If there are many distinguishable cases, we recommend having separate requirements - otherwise it is sufficient to complement the existing requirement.

It is important that you describe the system behavior for all cases - difficulties arise, in particular, when a simple case distinction with two categories is not sufficient. Thus, the analysis of a case distinction might lead to very complex requirements, whereby the limits of natural language are quickly reached. Suitable display methods to meet this challenge may be:

- bullet points for the individual cases
- decision tables, within which complex conditional structures can be depicted clearly.

Decision tables, in particular, help to check the completeness of a conditional structure by identifying variations of actions or conditions not described.

Rule 18: Analyze implicit assumptions in the requirement statement.

When describing a system, it happens very often that especially technically experienced stakeholders no longer communicate many relevant facts when requirements are being determined. Such facts are assumed to be known or considered too trivial to describe. The stakeholder is simply not aware of his extensive expertise.

Particularly for the sake of completeness, these inherent assumptions must be expatiated in the form of requirements. This is the only mean of preventing the deletion of information, without which the other statements or requirements do not make sense.

Descriptions often include statements that must be true in order for the description to make sense. Such statements are made using signal words, indicating inherent assumptions such as:

- temporal/logical process descriptions
- nouns that are more closely defined using a reference word

Check implicit assumptions by following the following steps:

1. Identify the signal word in the description.
2. Check whether the signal word refers to an implicit assumption.
3. Check whether this implicit assumption is defined by means of other, already-existing descriptions.
4. Explicitly define each implicit aspect that has not yet been described.

After the library system has saved the entered registration data of a new library user, the library system shall print a library card.

Nouns that are detailed by a reference word are indicators of implicit assumptions. In the example requirement, we note that the reference word “entered” before the noun “registration data” indicates that a function for entering registration data exists. If this has not yet been described by means of a requirement, information was deleted and you have uncovered an implicit assumption.

The SOPHIST Set of RExulations offers an approach that helps you to improve the quality of your requirements. On the next page we give you an overview of all of the rules in the Set of RExulations. For further information we recommend referring to [Rupp14].

Clear requirements, whose information has not been deleted, distorted or generalized, make further work on the basis of your requirements document easier. You create a better basis for contracts and facilitate communication between the stakeholders, thus requiring less potential revisions and fewer change requests. That saves you time and, thus, money.

Note, however, that you do not need to apply all 18 rules to each specification level in detail - on the upper levels of detail, you can already achieve significant improvements if you “only” eliminate nominalizations (Rule 3) and ask the wh-questions (“Who?”, “What?” and “Why?”) about the resulting main verbs. Familiarize yourself with the most important rules, apply them, and incorporate further rules into your repertoire step by step. You will thus continuously improve the quality of your requirements more and more.
5. Requirements Validation and Consolidation

Once your requirements are identified and documented, it is time to ensure, through the validation and consolidation of these requirements, that they comply with established quality criteria.

- **Validate requirements**

Faulty requirements hinder development activities. The later an error is detected, the more changes need to be undertaken - be it in the source code, the test artifacts or the architecture descriptions. Therefore, define quality criteria and assess your requirements thereafter using these criteria at an early stage. Find yourself a suitable testing method that matches your objectives and project circumstances. For more information about the validation of requirements and the metrics for evaluating requirements see [Rupp14].

- **Consolidate requirements**

In the context of the requirements engineering, disagreements can occur at many turns. These range from technical misunderstandings to severe personal conflicts, which cannot be resolved without additional help.

Conflicts that are dealt with as part of the RE process describe the incompatibilities of requirements based on a contradictory perception held by the respective stakeholders.

Once a conflict has been identified, it is the task of the requirements engineer to analyze it. Based on this analysis, it can be seen whether and how the conflict can be resolved together with the stakeholders involved. Following the resolution of a conflict, it is advisable to document the conflict, as well as the process and the result of the solution, in order to be able to work out a solution for similar conflicts consistently and speedily.
6. Requirements Management

Managing requirements, also known as requirements management, includes the processes that support you during the requirements analysis and further use of such requirements. But why would you want to deal at all with managing requirements?

Requirements change

As requirements change frequently in the course of a project, it is necessary that you find your way within your requirements specifications. Changes range from small repairs, such as spelling errors, to complex changes that include extensive revisions of entire sections of your specifications. A structured approach on how to deal with such so-called ‘change requests’ should be contained in your RE concept.

Requirements are reused

Always keep in mind that requirements are never collected for their own sake, but that stakeholders, such as developers or testers, have to be able to read, understand and work with your requirements. As a requirements engineer, you must therefore ensure that this comprehensive information is presented clearly in the specifications.

In the following sections we will give you an insight into the world of requirements management (abbreviated ‘RM’) as well as practical tips regarding the extent to which you should carry out certain activities belonging to requirements management.

6.1 How Much RM Makes Sense?

The importance of requirements management within the development process is directly related to the parameters of your project.

While there is no precise specification concerning how much RM you should plan in, the following factors can help you estimate the amount of RM needed:

- **The number of requirements and information**: The more information and requirements you collect, the more RM you are going to need.
- **The number of changes expected**: If you learn a lot about a product during the project, you can expect even more changes.
- **The estimated lifespan of the product**: If the product has a long life, it might entail many subsequent versions and maintenance work.
- **The persons involved in the RE process**: Too many cooks spoil the broth - this is also true in RM, where you have to clearly define and also adhere to your rights-role concept and your processes when the number of people involved increases.
- **The availability and involvement of the stakeholders**: The amount of RM necessary will increase if the stakeholders involved are difficult to reach - this can be seen, for example, when it comes to international projects.
- **The quality demands on the system**: Enhancing and ensuring the quality of a system usually goes hand-in-hand with more RM.
- **Reuse**: If other, similar products or even product lines are to be developed, additional RM can simplify and accelerate the subsequent development.
- **The complexity of the development process**.
- **Diverging stakeholder opinions**: Require many votes, often entailing changes - wide-ranging RM becomes necessary in order not to lose track.

The amount of requirements management needed should not be underestimated. However, it is hard to give an accurate forecast of the expected effort and cost in advance. Based on the above criteria, a rough guide to the expected amount of RM can, however, be identified. In the following section we will take a closer look at some of the most important aspects of RM.

Fig. 17: Good requirements management pays off...
6.2 Versioning and Baselines

As requirements change over the course of a project, in order to be able to understand how requirements have changed over time at a later date, the versioning of requirements has proven to be a useful tool.

When creating a new version of a requirement, the requirement is firstly copied. The old requirement is kept and linked to the new version. This new version is also given a new version number. The old requirement is entered into the history of your requirements specification. You can now edit the new version. This procedure ensures that no information gets lost.

Many tools developed specifically for RM support versioning. This is one of the reasons for using a professional RM tool.

Moreover, versioning also helps you to plan releases and changes to requirements. In doing so, one places requirements in an unalterable state, on which you can fall back at a later date. This selection of requirements is referred to as ‘configuration’.

If a configuration includes all the requirements for a release, we speak of a ‘baseline’ instead of a ‘configuration’. Give each configuration and baseline a unique name for identification purposes.

In order to find out which requirements are part of a configuration or a baseline, you need a traceability concept.

New version number = old version number +1

1. Copy requirement
2. Link old version with new version
3. Assign new version number

also known as “freeze”

6.3 Traceability

Traceability is the ability to find out relationships of requirements to each other, to defined objectives, but also to configurations and baselines. Only then can you find out, for instance when the aim has been changed, which requirements are affected by these changes, what requirements are needed for the development of a system functionality and which test cases cover these requirements.

Implementing traceability requires that requirements have attributes. Use the attribute values in order to get an overview of your requirements specifications by means of filtering and generating views.

Traceability provides the basis for effective and high-quality requirements management, as it supports the following aspects:

- **Verifiability**: Have all goals, requirements, test cases etc. been realized?
- **Identifying gold-plated solutions**: Which system functions have possibly not been required? Which requirements have no source or serve no fixed target?
- **Impact analysis**: What effect does the change of a goal have on the requirements that describe the goal?
- **Reuse**: Which artifacts from the development process can I use in other projects?
- **Accountability**: How much effort has gone into the realization of a goal, a use case or a single requirement?
- **Maintenance and care**: Root cause analysis, cost estimation for troubleshooting etc.

Traceability is the ability to trace connections and dependencies between information at any time during the analysis, implementation, maintenance and further development.
6.4 Change and Release Management

Frequent and complex changes to systems require a sophisticated approach to cover all pending processes - be it collecting change requests, planning of releases, or rolling out the changes. The appropriate techniques can be divided into the areas of change management and release management.

Change management controls the life cycle of all changes with the aim to minimize interruptions in running processes. The activities of change management include:
- prioritizing requests for change
- roughly describing change requests
- taking decisions on the implementation of the change.

Release management takes place once the planning, scheduling and control of builds and tests, as well as the integration into existing services is pending. The release manager has a special role in the process by monitoring the compliance of all appointments.

If the implementation is complete, the new system must go to the customer. The release depends on the respective project - a new product version can come out if many innovations were implemented, or it can also start a recall if you have eliminated a critical error. Do not forget to supply your service hotline, training staff, guide authors and all other relevant roles with the necessary information, e.g. in the form of release notes, in order to prepare them in time for the changes.

7. Conclusion

Requirements of a system form the core of successful system development. A project can succeed or fail thanks to them since they constitute the basis for contracts, proper implementation and development, tests and acceptance criteria.

As part of this booklet, we have tried to give you an idea of what the fundamental work of a requirements engineer entails. In doing so, we laid the focus on the elicitation and documentation of requirements in natural language, as this is the basis for the further work of a requirements engineer. The activities shown belong to the set of tools of a requirements analyst and help to increase the chances of successful project completion from the start.

Requirements engineering is a sophisticated field that opens up many possibilities. Professional RE thus boosts the chances of a project’s success, not only with regard to the IT environment, as it can be ensured by means of thorough analysis that:
- the system being developed is fully described
- the fulfillment of objectives is ensured by the associated requirements
- only validated and consolidated requirements, which serve the purpose of reaching the objectives, are implemented.

Although the initial cost may seem high - you will save a lot of money if you are developing in the right direction and according to the customer’s requirements from the very beginning. Requirements engineering pays off!
8. Sources


Benefits of a SOPHIST Training:

- Our trainers are professionally trained and CPRE certified.
- All our trainers acquire their competence through active project work and by solving actual problems occurring in the industry.
- Holistic processing of our trainings – from the organization and preparation to the follow-up of the training.
- Professional and profound training documents – on request also digital.
- Comfort version for our CPRE trainings – certification and double bottom with a free second chance*

Open Trainings

Our Open Trainings are aimed at individuals or smaller groups. Discuss with training participants from most different fields and expand your network.

Our Open Trainings are available at different locations in Germany. For further information, please see:

www.sophist.de/en/trainings/open-training/

Inhouse Trainings

You would like to train your whole team? Then choose a preconfigured SOPHIST Inhouse Training or create your own company-specific training together with SOPHIST!

Have a glance at our Training Portfolio:

www.sophist.de/en/trainings/inhouse-training/

*Free repetition of the specific training + certification if you have booked an Open Training. If you have attended an Inhouse Training, you can visit an Open Training + certification for 50% of the price.
Das neue Trainingsformat mit schnellerem Lernerfolg und mehr Praxisbezug!

Requirements-Engineering in der Praxis

- Lernen Sie die Grundlagen des Requirements-Engineering und testen Sie Ihr Wissen ortsungebunden mit Hilfe unserer E-Learning Plattform im Vorfeld.
- An den 2 Tagen der Präsenzveranstaltung werden die Techniken der Anforderungsanalyse und deren Einsatz in Ihrem Projekt erläutert.
- Der Trainer geht gezielt auf Ihre Ergebnisse der E-Learning Plattform und Ihre Fragen mit viel Praxisbezug ein.

Hier finden Sie weitere Informationen:
www.sophist.de/bl-o-reg/

Deshalb gleich die neuen Trainings buchen:

Certified Professional for Requirements Engineering Foundation Level

- Lernen Sie die theoretischen Inhalte für die Zertifizierung und testen Sie Ihr Wissen ortsungebunden mit Hilfe unserer E-Learning Plattform im Vorfeld.
- An den 2 Tagen der Präsenzveranstaltung steht die Vorbereitung auf die Prüfung im Mittelpunkt.
- Der Trainer geht gezielt auf Ihre Ergebnisse der E-Learning Plattform und Ihre Fragen mit viel Praxisbezug ein.
- Sie erhalten zudem ausführliche Seminarunterlagen, das Buch „Basiswissen Requirements-Engineering“ und vieles mehr!

Hier finden Sie weitere Informationen:
www.sophist.de/bl-cpre-fl/
The basics of requirements engineering

When developing a system - be it software or a building - the most important prerequisite for success is that all those who are involved know exactly what is to be developed. And this already begins with the rough targets and ends with highly detailed instructions for the implementation.

The work of requirements engineers deals precisely with this topic. Their job involves:

• elicitation,
• documentation,
• validation and consolidation, as well as
• management

of system requirements.

With this brochure, we want to give you an overview of the activities belonging to these roles and, hence, what we, the SOPHISTs, have specialized in for now over 20 years.