# 2 System and Context Boundaries

The requirements for a system to be developed do not simply exist, they have to be elicited. The purpose of defining the system and context boundaries in requirements engineering is to identify the part of the environment that influences the requirements for the system to be developed.

## 2.1 System Context

*Anticipate the system in operation*

In the development process, requirements engineering fulfils the task of identifying all those material and immaterial aspects that have a relationship to the system. In order to do that, it is anticipated what the system will be like once it becomes real. By doing so, those parts of the real world which will potentially influence the requirements of the system can be identified. To be able to specify the requirements for a system correctly and completely, it is necessary to identify the relationships between individual material and immaterial aspects as precisely as possible. The part of reality that is relevant for the requirements of a system is called the system context.

**Definition 2-1:** *System Context*

The system context is the part of the system environment that is relevant for the definition as well as the understanding of the requirements of a system to be developed.

*Context aspects in the system context*

Among others, the following possible aspects of reality influence the context of a system:

- People (stakeholders or groups of stakeholders)
- Systems in operation (other technical systems or hardware)
- Processes (technical or physical processes, business processes)
- Events (technical or physical)
- Documents (e.g., laws, standards, system documentation)

*Consequence of erroneous or incomplete context consideration*

If the system context is incorrectly or incompletely considered during requirements engineering, it may result in incomplete or erroneous requirements. This leads to the system operating on the basis of incomplete or erroneous requirements, which is often the reason for system failure during operation. Such errors often remain undetected during the validation procedures, which determine if the system meets the specified requirements, and occur only during operation, sometimes entailing catastrophic consequences.

*System context and requirement context*

The origin of the system's requirements lies within the context of the system to be developed. For example, stakeholders, pertinent standards, and legal guidelines demand particular functional properties that the system to be developed must possess at its interfaces. A requirement is therefore defined for a specific context and can only be interpreted correctly in regard to this specific context. The better the context of a requirement is understood (e.g., why is the technical system "X" in the system context the origin of some requirement), the lower the likelihood of incorrect interpretation of the requirement. Therefore, a purpose-driven documentation of the system context or information about the system context is of particular importance.

## 2.2 Defining System and Context Boundaries

It is within the responsibility of the requirements engineer to define the system context properly. In order to do so, it is necessary to separate the system context from the system to be developed as well as from the parts of reality that are irrelevant for the system (see figure 2-1):

- *Defining the system boundary:* When defining the system boundary, a decision has to be made: Which aspects pertain to the system to be developed and which aspects belong in the system context?
- *Defining the context boundary:* When defining the context boundary, the question to be answered is: Which aspects pertain to the system context (i.e., have a relation to the system to be developed) and which aspects are part of the irrelevant environment?

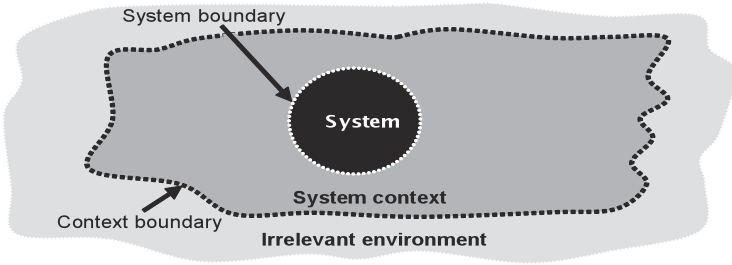


***Figure 2-1*** *System and context boundary of a **s**ystem*

Thus, system and context boundaries define the system context. The system context comprises all aspects that are relevant with regard to the requirements for the system to be developed. These aspects cannot be altered or modified by the system development process.

*System and context boundaries define the system context.*

### 2.2.1 Defining the System Boundary

The system boundary separates the object of concern (i.e., the system) from its environment. When the system boundary is defined, the scope of the development (i.e., the aspects that are covered by the system to be developed) as well as the aspects that are not part of the system are determined. We therefore define the system boundary as follows:

> **Definition 2-2:** *System Boundary*
>
> The system boundary separates the system to be developed from its environment; i.e., it separates the part of the reality that can be modified or altered by the development process from aspects of the environment that cannot be changed or modified by the development process.

All aspects that are within the system boundary can thus be altered during system development. For instance, an existing system that consists of hardware and software components and is supposed to be replaced by the new system can be within the system boundary. Aspects within the system context can be business processes, technical processes, people and roles, organizational structures, and components of the IT infrastructure. Figure 2-2 schematically shows the system context of a system. The system context consists of other systems, groups of stakeholders that in some way use the interfaces of the system to be developed, and additional requirements sources and their interrelations.

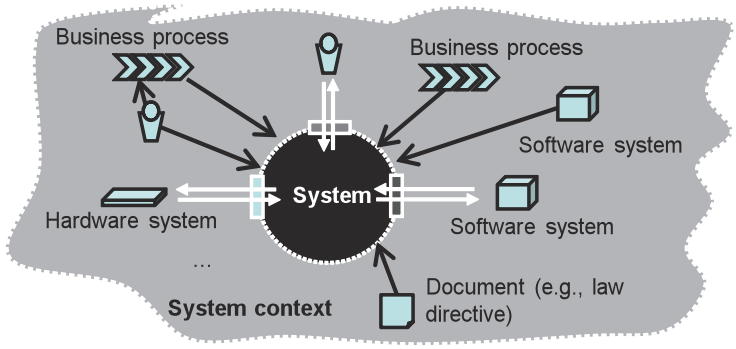


***Figure 2-2*** *Types of aspects within the system context*

*Sources and sinks as the starting point*

Among other things, sources and sinks (see, e.g., [DeMarco 1978]) can be used to identify the interfaces the system has with its environment. Sources provide inputs for the system. Sinks receive outputs from the system. Possible sources and sinks of a system are as follows:

- (Groups of) stakeholders
- Existing systems (both technical and nontechnical systems)

*Interfaces: interaction between system and environment*

Sources and sinks interact with the system to be developed via system interfaces. Using these interfaces, the system provides its functionality to the environment, monitors the environment, influences parameters of the environment, and controls operations of the environment. Depending on the type of the respective source or sink, the system needs different interface types (e.g., human–machine interface, hardware interface, or software interface). The interface type in turn may also impose specific constraints or additional sources of requirements on the system to be developed.

*Gray zone between system and system context*

Frequently, the system boundary is not precisely defined until the end of the requirements engineering process. Before that, some or several interfaces as well as desired functions and qualities of the system to be developed are only partially known or not known at all. We refer to this initially vague separation of the system and its context as the gray zone between the system and the context (see figure 2-3). At the beginning of the requirements engineering process, it may, for example, not be clear whether the system should implement a certain function (e.g., “pay by credit card”) or whether there is another system in the system context providing such a function that should be used (e.g., “payment processing”).

*Adjusting the gray zone*

The system boundary may not only shift within the gray zone (① in figure 2-3) but also the gray zone itself may shift during the requirements engineering process (② in figure 2-3). This kind of shifting is caused by the fact that aspects, pertaining at first to the system context, now will be modified during system development. Such a situation occurs during requirements engineering, for example, if it is not clear in the system context whether certain activities of a business process should be implemented or supported by the system to be developed or not. In this situation, it is not clear which aspects belong to the system and can thus be changed or modified and which aspects belong to the system context. This causes a corresponding shift of the gray zone between system and system context (see figure 2-3).

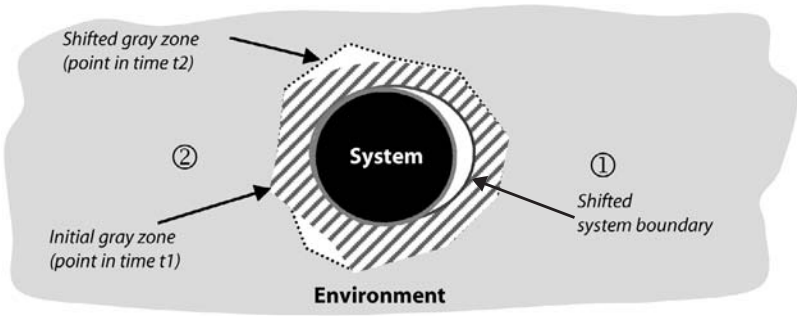


***Figure 2-3*** *Gray zone of the system boundary*

The gray zone shifts, for instance, when interfaces are attributed to the system boundary and the gray zone is extended to comprise aspects of the environment that concern these interfaces.

### 2.2.2 Defining the Context Boundary

The context boundary distinguishes between context aspects, i.e., those aspects of the environment that need to be taken into account during requirements engineering (e.g., as requirements sources) and those aspects that are irrelevant for the system. The context boundary can be defined as follows:

**Definition 2-3:** *Context Boundary*

The context boundary separates the relevant part of the environment of a system to be developed from the irrelevant part, i.e., the part that does not influence the system to be developed and, thus, does not have to be considered during requirements engineering.

*Concretion and shift of the context boundary*

At the beginning of the requirements engineering process, frequently only part of the environment as well as single specific relationships between the environment and the system to be developed are known. In the course of requirements engineering, it is necessary to concretize the boundary between system context and irrelevant environment by analyzing relevant aspects within the environment with regard to their relationships to the system. Besides the system boundary, the context boundary typically also shifts during requirements engineering. For instance, it may be possible that a law directive that was considered to be relevant for the system to be developed no longer impacts the system or is no longer considered relevant. The system context is therefore reduced (① in figure 2-4). If a new law directive is identified that influences the system, the system context is extended accordingly (② in figure 2-4).

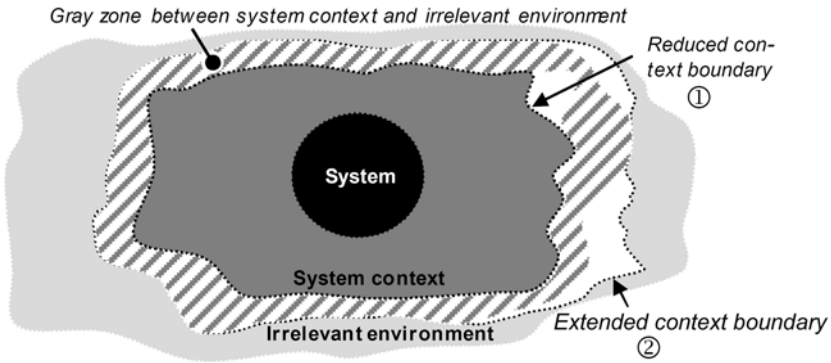


***Figure 2-4*** *Gray zone between system context and irrelevant environment*

*Gray zone between system context and irrelevant environment*

Since the context boundary separates the system context from those parts of reality that are irrelevant to the system, a complete and precise definition of the context boundary for complex systems is virtually impossible. In addition, it may not be possible to clarify for single aspects of the environment whether they influence the system to be developed or are influenced

by it or not. These two observations are the reason for the existence of a gray zone with regard to the context boundary (see figure 2-4 ).

*Resolving and shifting of the gray zone*

This gray zone therefore comprises identified aspects of the environment for which it is unclear whether they have a relation to the system or not. In contrast to the gray zone between the system and the system context that must be resolved in the course of requirements engineering, it is not necessary to resolve the gray zone between the system context and the irrelevant environment entirely.

## 2.3 Documenting the System Context

In order to document the system context (especially the system and context boundaries), "use case" diagrams [Jacobson et al. 1992] (see sections 4.2.3 and 6.3.1) or "data flow" diagrams [DeMarco 1978] (see section 6.6.1) are often used. When the context is modeled with data flow diagrams, sources and sinks in the environment of the system that represent the source or destination of data flows (or flows of material, energy, money, etc.) are modeled. In use case diagrams, actors (such as people or other systems) in the system environment and their usage relationships to the system are modeled. To model the system context, UML class diagrams [OMG 2007] (see section 6.5.2) may also be used. In order to document the system context of a system as thoroughly as possible, typically several documentation forms are used.

## 2.4 Summary

The system context is the part of the reality that influences the system to be developed and thus also influences the requirements for the system. In order to be able to elicit the requirements for the system to be developed, it is necessary to define the boundary of the system to the system context and the boundary of the system context to the irrelevant environment first. When the system boundaries are defined, the scope of the system is determined. The scope comprises those aspects that can be changed and designed during system development. At the same time, it is also defined which aspects belong to the environment and thus cannot be altered during development and may provide constraints for the system to be developed.

The context boundary separates the part of the environment that influences the requirements for the system to be developed from that part that does not influence the requirements. Typical aspects within the system context are stakeholders (e.g., the users of the system) and documents (e.g., standards that have to be considered) as well as other systems that, for instance, interact with the system to be developed. Defining the system and context boundaries successfully is the foundation for a systematic elicitation of requirements for the system to be developed.

# 4 Documenting Requirements

In requirements engineering, information that has been established or worked out during different activities must be documented. Among this information are, for example, protocols of interviews and reports of validation or agreement activities, but also change requests. The main and most important documentation task in requirements engineering, though, is to document the requirements for the system in a suitable manner.

## 4.1 Document Design

A documentation technique is any kind of more or less formal depiction that eases communication between stakeholders and increases the quality of the documented requirements. In principle, any kind of documentation technique can be used to document the requirements, let it be natural language-based documentation by means of prose, more structured natural language-based text, or more formal techniques such as state diagrams.

**Definition 4-1:** *Requirements Document / Requirements Specification*

A requirements specification is a systematically represented collection of requirements, typically for a system or component, that satisfies given criteria.

*Reasons for the documentation*

During the life cycle of a requirements document, many people are trusted with the documentation. During communication, the documentation has a goal-oriented and supporting role. The main reasons for documenting requirements are as follows:

*Central role of requirements*

- *Requirements are the basis of the system development.* Requirements of any kind influence the analysis, design, implementation, and test phases directly and indirectly. The quality of a requirement or of a requirements document has a strong impact on the progress of the project and therefore on its success.

*Legal relevance*

- *Requirements have a legal relevance.* Requirements are legally binding for the contractor and the client, and the client can sue for their fulfillment. Documenting the requirements can help to quickly overcome legal conflicts between two or more parties.

*Complexity*

- *Requirements documents are complex.* Systems that possess thousands of requirements that in turn have complex interdependencies on multiple layers are not unheard of in practice. Without suitable documentation, keeping on top of things can become very difficult for anyone involved.

*Accessibility*

- *Requirements must be accessible to all involved parties.* Projects undergo certain "development" as time goes by—with regard to the subject as well as the staff. When requirements can be permanently accessed, uncertainty and obscurities can be avoided and staff that has recently joined the project can quickly get up to speed.

Another argument for a good documentation, supportive of the project, is that employees almost never share the same understanding of a subject matter. Therefore, requirements should be documented in a way that they meet the quality demands of all involved.

## 4.2 Types of Documentation

Requirements for a system can be documented in three different perspectives. In practice, natural language as well as conceptual models are used to this end, or oftentimes, an advantageous combination of both is employed.

### 4.2.1 The Three Perspectives of Requirements

Requirements for a system can be documented in three different perspectives onto the system to be developed:

*Data perspective*

- *Data perspective:* In the data perspective, a static-structural perspective on the requirements of the system is adopted. For example, the structure of input and output data as well as static-structural aspects of usage and dependency relations of the system and the system context can be documented (e.g., the services of an external system).

*Functional perspective*

- *Functional perspective:* The functional perspective documents which information (data) is received from the system context and manipulated

by the system or one of its functions. This perspective also documents which data flows back into the system context. The order in which functions processing the input data are executed is also documented.

- *Behavioral perspective:* In the behavioral perspective, information about the system and how it is embedded into the system context is documented in a state-oriented manner. This is done by documenting the reactions of the system upon events in the system context, the conditions that warrant a state transition, and the effects that the system shall have on its environment (e.g., effects of the system analyzed that represent events in the system context of a different system).

*Behavioral perspective*

### 4.2.2 Requirements Documentation using Natural Language

*Advantages of using natural language*

Natural language, particularly prose, is the most commonly applied documentation form for requirements in practice. In contrast to other documentation forms, prose has a striking advantage: No stakeholder has to learn a new notation. In addition, language can be used for miscellaneous purposes—the requirements engineer can use natural language to express any kind of requirement.

*Disadvantages of using natural language*

Natural-language-based documentation is well suited to document requirements in any of the three perspectives. However, natural language can allow requirements to be ambiguous, and requirements of different types and perspectives are in danger of being unintentionally mixed up during documentation. In that case, it is difficult to isolate information pertaining to a certain perspective amidst all of the requirements in natural language.

### 4.2.3 Requirements Documentation using Conceptual Models

In contrast to natural language, the different types of conceptual models cannot be used universally. When documenting requirements by means of models, special modeling languages must be used that pertain to the appropriate perspective. Assuming the modeling language selected for a documentation task is applied correctly, its use constructively guarantees that the models created depict information pertaining to the respective perspective only. The models depict the documented requirements much more compactly and they therefore are easier for a trained reader to understand than is natural language. In addition, conceptual models offer a decreased degree of ambiguity (i.e., fewer ways to be interpreted) than

natural language due to their higher degree of formality. However, using conceptual modeling languages for requirements documentation requires specific knowledge of modeling. The following list includes short descriptions of the most important diagrams discussed in chapter 6.

*Overview of system functions*

- *Use case diagram:* A use case diagram allows you to gain a quick overview of the functionalities of the specified system. A use case describes which functions are offered to the user by the system and how these functions relate to other external interacting entities. However, use cases do not describe the responsibilities that the functions have in detail (see section 6.3).

*Structural data modeling and structuring of terms*

- *Class diagram:* Among other things, class diagrams are used in requirements engineering to document requirements with regard to the static structure of data, to document static-structural dependencies between the system and the system context, or to document complex domain terms in a structured manner (see section 6.5.2).

*Sequence modeling*

- *Activity diagram:* Using activity diagrams, business processes, or sequence-oriented dependencies of the system in regard to processes within the system context can be documented. Activity diagrams are also well suited to model the sequential character of use cases or to model a detailed specification of the interaction of functions that process data (see section 6.6.3).

*Event-driven behavior*

- *State diagram:* State diagrams are used in requirements engineering to document event-driven behavior of a system. The focus of this type of model is on the individual states the system can be in, events and their respective conditions that trigger a state transition, and effects of the system in its environment.

### 4.2.4 Hybrid Requirements Documents

*Combined use of documentation types*

Requirements documents first and foremost contain requirements. In addition, in many situations it is sensible to document decisions, important explanations, and other relevant information as well. Depending on the target audience of the document, the perspective on the system, and the documented knowledge, suitable documentation types are selected. Typically, documents contain a combination of natural language and conceptual models. The combination allows the disadvantages of both documentation types to be decreased by means of the strengths of the other documentation type, and combining documentation types exploits the

advantages of both. For instance, models can be amended or complemented by natural language comments and natural language requirements and natural language glossaries can be summarized and their dependencies can be depicted clearly by making use of models.

## 4.3 Document Structures

*Influence of the requirements on satisfaction*

Requirements documents contain a magnitude of different information. These must be well structured for the reader. In order to do that, one can make use of standardized document structures or individually define a custom document structure.

### 4.3.1 Standardized Document Structures

*Adaptation of existing standard outlines*

Standard outlines offer a predefined structure, i.e., predefined stereotypes according to which the information can be classified. By using standard outlines, a rough structure along with a short description of the content of the main sections is predetermined. Using standard outlines has the following advantages:

- Standard outlines simplify incorporating new staff members.
- Standard outlines allow for quickly finding desired contents.
- Standard outlines allow for selective reading and validation of requirements documents.
- Standard outlines allow for automatic verification of requirements documents (e.g., with regard to completeness).
- Standard outlines allow for simplified reuse of the contents of requirements documents.

It must be noted that these structures must be tailored with regard to the specific project properties to meet the respective constraints. In the following paragraphs, three of the most widely used standardized document structures are introduced.

*Rational Unified Process*

The *Rational Unified Process (RUP)* [Kruchten 2001]is usually used for software systems that are developed using object-oriented methods. The client creates a *business model* that contains different artifacts from the business environment (e.g., business rules, business use cases, business goals), which serve as the basis for requirements of the system over the course of development. The contractor uses the structures of the *software*

*requirements specification (SRS)* to document all software requirements. These structures are closely related to the ISO/IEC/IEEE standard 29148:2011, as described next.

*ISO/IEC/IEEE standard 29148:2011*

The ISO/IEC/IEEE standard 29148:2011 [ISO/IEC/IEEE 29148:2011] contains an outline designed for the documentation of software requirements (software requirements specification). The standard structure suggests dividing the requirements document into five parts with regard to their subject matter:

- A chapter with introductory information (e.g., system goal, system bounding) and a general description of the software (e.g., perspective of the system, properties of future users)
- A chapter with a listing of all documents that are referenced in the specification
- A chapter for specific requirements (e.g., functional requirements, performance, interfaces)
- A chapter with all planned measures for verification
- Appendices (e.g., information about assumptions that were made, identified dependencies)

*V-Model*

The *V-Model* [V-Modell 2004] of the German Federal Ministry of the Interior (BMI) defines different structures, depending on the creator of the requirements document:

- The *Customer Requirements Specification*, known in the German original as Lastenheft, is created by the customer and describes all of the demands to the contractor regarding the subject of the contract, i.e., deliveries and services. In addition, in many cases, demands of the users, including all constraints to the system and the development process, are documented. Therefore, the Customer Requirements Specification usually describes *what* is made *for what*.
- The *System Requirements Specification*, known in the German original as Pflichtenheft, is based on the Customer Requirements Specification and contains the implementation suggestions that the contractor has elaborated. Therefore, the System Requirements Specification is a refinement of the requirements and constraints of the Customer Requirements Specification.

### 4.3.2 Customized Standard Contents

*The minimum content*

As described in section 4.3.1, standardized document structures are adapted with regard to the specific project conditions. The following issues should be addressed by any chosen structure.

#### Introduction

The introduction contains information about the entire document. This information allows gaining a quick overview of the system.

- *Purpose:* This section discusses why the document was created and who the target audience for the requirements document is.
- *System coverage:* This part consists of the system to be developed. It indicates system name and the principle goals and advantages that arise from introducing the system.
- *Stakeholder:* This section contains a list of stakeholders and their relevant information (see section 3.1.1).
- *Definitions, Acronyms, and Abbreviations:*[1] In this section, the terms used in the document are defined so that they can be used consistently throughout the document.
- *References:*[2] All documents that are referenced by the requirements document are listed herein.
- *Overview:* At the end of the introductory chapter, the content and structure of the following sections of the requirements document should be explained briefly.

#### General Overview

In this section, additional information is documented that increases the understandability of the requirements. In contrast to the introduction, this is merely operational information that does not pertain to administration, management, or organizational aspects of the requirements document.

- *System environment:* The embedding of the system into the environment is of key concern in this paragraph. The results of your definition of the system boundary and context boundary can be found herein.
- *Architecture description:* In this section, the operational interfaces of the system (e.g., user interfaces, hardware and software interfaces, and

1. This section can also be treated as an appendix to the document.
2. This section can also be treated as an appendix to the document.

communication interfaces) are documented. In addition, further information, e.g., regarding storage limitations, is also discussed.
- *System functionality:* This section contains the coarse functionalities and tasks of the system. This can be documented, for example, using a use case diagram.
- *User and target audience:* The different users of the system that make up the target audience are listed.
- *Constraints:* In this section, all conditions ought to be listed that have not been documented thus far and might hinder the requirements engineering.
- *Assumptions:* Decisions, such as not implementing certain aspects of the system due to budgeting reasons, or other general assumptions about the system context that the requirements are based upon are documented here.

#### Requirements

This part contains functional requirements as well as quality requirements.

#### Appendices

In the appendices, additional information that completes the document can be documented. For example, the appendices can include additional documents regarding the user characteristics, standards, conventions, or background information regarding the requirements document.

#### Index

The index typically contains a table of contents (i.e., a structure of the chapters) and an index directory. In highly dynamic requirements documents, this may be a highly critical section that must be kept up-to-date.

## 4.4 Using Requirements Documents

*Requirements documents as the basis for development*

Over the course of the project, requirements documents serve as the basis for different tasks:

- *Planning:* Based on the requirements document, concrete work packages and milestones for the implementation of the system can be defined.

- *Architectural design:* The detailed documented requirements (along with constraints) serve as the basis for the design of the system architecture.
- *Implementation:* Based on the architectural design, the system is implemented by making use of the requirements.
- *Test:* On the basis of requirements that have been documented in the requirements document, test cases can be developed that can be used for system validation later on.
- *Change management:* When requirements change, the requirements document can serve as the basis to analyze the extent to which other parts of the system are influenced. The change effort can thus be estimated.
- *System usage and system maintenance:* After the system is developed, the requirements document is used for maintenance and support. This way, the requirements document can be used to analyze concrete defects and shortcomings that surface during system use. For example, one can deduct if a defect is a result of using the system incorrectly, a result of an error in requirements, or a result of an error in implementation.
- *Contract management:* The requirements document is the prime subject of a contract between a client and a contractor in many cases.

## 4.5 Quality Criteria for Requirements Documents

To become a basis for the subsequent processes, the requirements document must meet certain quality criteria. According to the ISO/IEC/IEEE standard 29148:2011 [ISO/IEC/IEEE 29148:2011], a requirements document shall be complete and consistent. Moreover, a requirements document shall support readability by offering a clear structure, reasonable scope, and traceability. Overall, a requirements document shall fulfil the following quality criteria:

- Unambiguity and consistency
- Clear structure
- Modifiability and extendibility
- Completeness
- Traceability

### 4.5.1 Unambiguity and Consistency

*Quality of individual requirements is a prerequisite.*

Requirements documents can be consistent and unambiguous only when the individual requirements are consistent and unambiguous. In addition, it must be guaranteed that individual requirements do not contradict one another. To achieve this, it is advisable to make use of conceptual models (see chapter 6). Another aspect of unambiguity pertains to the unique identification of a requirements document or a requirement among the set of all requirements documents or requirements in a development project (see section 8.5).

### 4.5.2 Clear Structure

*Allows for selective reading*

In order to guarantee that the requirements document is readable by any stakeholder, it should be appropriately comprehensive and clearly structured. Unfortunately, no clear-cut suggestions can be made regarding the appropriate comprehensiveness of a requirements document. A very comprehensive requirements document with a good structure can be just as appropriate as a less comprehensive document because a clear structure will allow the reader to skip parts that are not relevant to him. An unstructured or badly structured requirements document of the same high level of comprehensiveness would not be appropriate because the document must be read in its entirety in order for a stakeholder to be able to identify parts that are relevant to her. A good starting point is the standard structures described in section 4.3.1.

### 4.5.3 Modifiability and Extendibility

*Content and structure should support changeability.*

Requirements documents must be easy to extend. There are always requirements that are changed, altered, added, or removed as a project progresses. As a result, the structure of requirements documents should be easy to modify and extend. The requirements documents of a project should be subject to the project's version control management.

### 4.5.4 Completeness

*Two types of completeness in requirements documents*

Requirements documents must be complete, i.e., they must contain all relevant requirements (and required additional information), and each requirement must be documented completely.[3] All possible inputs, influ-

ential factors, and required reactions of the system must be described for each desired system function. This comprises describing error and exception cases in particular. Also, quality requirements, such as requirements pertaining to reaction times or availability and usability of the system, must be noted.

*Evidence, reference, and sources are formal necessities.*

Formal factors also contribute to completeness. Graphs, diagrams, and tables should be appropriately labeled. Another important aspect is that consistent reference and index directories must exist. Also, definitions and norm reference that denote specific terms must be included in any requirements document. The comprehensiveness of a requirements document is a challenge during requirements engineering. Often, a compromise must be found between the time resources available and the completeness of the requirements documents.

### 4.5.5 Traceability

*Relationship to other development documents*

An important quality criterion is traceability of relationships between requirements documents and other documents (e.g., business process model, test plans, or design plans). These documents could have been created in previous development phases, in subsequent development phases, or concurrently with the requirements documents. Among other things, traceability supports change management (see section 8.4).

## 4.6 Quality Criteria for Requirements

*Quality criteria for single document requirements*

Each documented requirement should fulfil the following quality criteria:

- *Agreed*: A requirement is agreed upon if it is correct and necessary in the opinion of all stakeholders.
- *Unambiguous*: [ISO/IEC/IEEE 29148:2011] A requirement that is unambiguously documented can be understood in only one way. It must not be possible to interpret the requirement in a different way. All readers of the requirement must arrive at the same understanding of the requirement.
- *Necessary*: [ISO/IEC/IEEE 29148:2011] A documented requirement must represent the facts and conditions of the system context in a way

3. Strictly speaking, this statement holds true only for the requirements document of the next system release (see section 8.5.3).

that it is valid with regard to the actualities of the system context. These actualities may be the different stakeholders' ideas, relevant standards, or interfaces to external systems.

- *Consistent*: [ISO/IEC/IEEE 29148:2011] Requirements must be consistent with regard to all other requirements, i.e., the requirements must not contradict one another, regardless of their level of detail or documentation type. In addition, a requirement must be formulated in a way that allows for consistency with itself, i.e., the requirement may not contradict itself.
- *Verifiable*: [ISO/IEC/IEEE 29148:2011] A requirement must be described in a way that allows for verification. That means that tests or measurements can be carried out that provide evidence of the functionality demanded by the requirement.
- *Feasible*: [ISO/IEC/IEEE 29148:2011] It must be possible to implement each requirement given the organizational, legal, technical, or financial constraints. This means that a member of the development team ought to be involved in rating the goals and requirements so that he can show the technical limits of the implementation of a particular requirement. In addition, the costs for the implementation must be incorporated into the rating. Occasionally, stakeholders withdraw a requirement if the costs for its realization become apparent.
- *Traceable*: [ISO/IEC/IEEE 29148:2011] A requirement is traceable if its origin as well as its realization and its relation to other documents can be retraced. This can be done by means of unique requirement identifiers. Using these unique identifiers, requirements that are derived from other requirements on a different level of the specification can be connected. For example, a system goal can be traced through all levels of abstraction, from design to implementation and test. Details can be found in section 8.4.
- *Complete*: [ISO/IEC/IEEE 29148:2011] Each individual requirement must completely describe the functionality it specifies. Requirements that are yet incomplete must be specially marked, for example by inserting "tbd" ("to be determined") into the respective text field or by setting a corresponding status. These markings can then be systematically searched for and missing information can be amended accordingly.
- *Understandable*: Requirements must be comprehensible to each stakeholder. Therefore, the type of requirements documentation (see sec-

tion 4.2) can vary significantly, depending on the development phase (and therefore, depending on the involved staff). In requirements engineering, it is important to strictly define the terms used.

*Fundamental principles of understandability*

Along with quality criteria for requirements, there are two fundamental rules that enhance the readability of requirements:

- *Short sentences and short paragraphs:* As human short-term memory is very limited, circumstances that belong together should be described in no more than seven sentences.
- *Formulate only one requirement per sentence:* Formulate requirements using active voice and use only one process verb. Long, complicated interlaced sentences must be avoided.

## 4.7 Glossary

A frequent cause for conflicts in requirements engineering is that the people that are involved in the development process have different interpretations of terms. In order to avoid these conflicts, it is necessary that everyone who is involved in the development process shares the same understanding of the terminology used. Therefore, all relevant terms must be defined in a common glossary. A glossary is a collection of term definitions and contains the following elements:

- Context-specific technical terms
- Abbreviations and acronyms
- Everyday concepts that have a special meaning in the given context
- Synonyms, i.e., different terms with the same meaning
- Homonyms, i.e., identical terms with different meanings

*Consistent definitions*

By defining the meaning of terms, you can increase the understandability of requirements considerably. Misunderstandings and different interpretations of terms that might lead to conflicts can be avoided from the beginning.

*Reuse of glossary entries*

Often, in different projects, terms are used that are similar to one another or in fact identical. This may be the case, for example, when one system is developed for different customers but within the same domain. In this case, already existing glossary entries should be reused. It may even be feasible to define such terms in a universal, inter-project glossary. The additional effort of creating such a glossary will pay off in future projects.

For certain domains, collections of term definitions already exist and are publicly accessible. These may serve as the foundation for the definition of specific glossaries. For example, in [IEEE 610.12-1990], typical terms of software engineering are defined.

#### Rules for Using a Glossary

*Basic rules for using a glossary*

Since creating a glossary is absolutely mandatory, the following must be noted:

- *The glossary must be centrally managed:* At any time, there must be only one valid glossary, which must also be centrally accessible. There must not be multiple valid glossaries.
- *Responsibility must be assigned:* One particular individual must be assigned with the task of maintaining the glossary and ensuring consistency and up-to-dateness. The necessary resources to accomplish this task must be included in the project plan.
- *The glossary must be maintained over the course of the project:* In order to ensure that the glossary is consistent and up-to-date, it must be maintained over the course of the entire project by the person that was assigned this responsibility.
- *The glossary must be commonly accessible:* The term definitions must be available for all involved personnel. This is the only way a common understanding of the terms can be ensured.
- *Using the glossary must be obligatory:* All involved personnel must be obliged to exclusively use the terms and term definitions as they have been defined in the glossary.
- *The glossary should contain the sources of the terms:* In order to be able to resolve questions and problems at any time during the course of the project, it must be possible to determine the source of a term.
- *The stakeholders should agree upon the glossary:* Only stakeholders can reliably validate the operational definitions for their respective project context. Each definition should be validated by the stakeholders or their representatives. In addition, the individual term definitions in the glossary should be explicitly approved. This approval signals that the respective term is correct and its use is obligatory.
- *The entries in the glossary should have a consistent structure:* All entries in the glossary must be structured in the same way. In order to support a consistent documentation, it is advisable to use a template for glossary definitions. In addition to the definition and the

meaning of a term, the template should specify possible synonyms and homonyms.

To reduce the effort of aligning terms with one another, it is advisable to start with the creation of the glossary early on in the project.

## 4.8 Summary

The documentation of requirements plays a central role in requirements engineering. As the amount of requirements is often vast, it is very important to clearly structure the requirements so that personnel not involved with the project also understand them. Also, looking up and changing requirements is simplified and accelerated in this way. This makes meeting the quality criteria for requirements documents much easier. Using customized documentation structures has proven to be suitable for that purpose. These are completed by inserting project-specific requirements written in natural language in conjunction with conceptual requirements models.