Chapter 2

Requirements

A *requirement* is a textual description of system behaviour. A requirement describes in plain text, usually English, what a system is expected to do. This is a basic technique much used in practice. The content of this chapter is based on the ISO/IEC/IEEE standard 29148:2011. You should be able to access the standard form the TU/e library. A key section of the standard is ”5.2 Requirements fundamentals”. The teacher shall never assume that you have read this standard but shall strongly encourage you to actually read it.

Note that requirements can be used all along the development life cycle. In this chapter, we consider the very initial phase of a software development project. We assume requirements are the first model of the system to be developed.

2.1 Learning objectives of this chapter

The learning objectives of this chapter are the followings. At the end of this chapter, you shall:

- know what is a well-defined requirement;
- know what syntax can be used to write well-formed requirements;
- know what is a well-defined set of requirements;
- be capable of writing a well-formed set of well-formed requirements.

2.2 Definition and syntax

A requirement is a *statement* expressing a need and its associated constraints and conditions. A requirement clearly states something that the system is expected to do. A requirement is written in natural language. When using a structure language with clearly defined syntax and vocabulary, requirements are said to be *semi-formal*. In this course, we will only consider semi-formal requirements. A requirement comprises at least:

- a subject: what is the subject of the requirement?
- a verb: what shall be done by the subject?

The ISO 29148 standard describes a clear syntax for writing requirements:

[Condition][Subject][Action][Object][Constraint]

Each part of the syntax has the following intention:

- **[Condition]**: condition under which the requirement applies. A condition will generally start with the word ”When ...”. For instance, ”When signal X is received” or ”When the system is in mode Y”.
- **[Subject]**: describes the *actor*. For instance ”the application” or ”the system”, etc.
- **[Action]**: describes the actual action of the requirement. For instance ”shall return to”, ”shall send”, etc.
- **[Object]**: the object of the action. This can be a signal, a message, a particular state, etc.
- **[Constraint]**: a constraint restricts the action. For instance, by giving a time limit. A typical word for constraints is ”within”.

Any requirements shall have a Subjet and an Action. The other parts are optionals. Let’s have a look at some examples (taken from the standard).

Example 2.2.1. When signal x is received **[Condition]**, the system **[Subject]** shall set **[Action]** signal x received bit **[Object]** within 2 seconds **[Constraint]**.

This example illustrates the difference between **[Condition]** and **[Constraint]**. A condition is a restriction on the subject or its environment. In the example, the condition that a particular signal is received. A constraint is a restriction on the action. In the example, that a change in a bit must take place within a given time limit. This difference is also illustrated by the following example:

Example 2.2.2. At sea state 1 **[Condition]**, the Radar System **[Subject]** shall detect **[Action]** targets **[Object]** at ranges out to 100 nautical miles **[Constraint]**.

Finally, here is an example without a condition:

Example 2.2.3. The invoice system **[Subject]** , shall display **[Action]** pending customer invoices **[Object]** in ascending order in which invoices are to be paid **[Constraint]**.

2.3 Requirements goal and characteristics

Now that we know what actually is a requirement, we can look at the goal of a set of requirements. Once we understand the objective of writing requirements, we will discuss the expected characteristics of requirements.

2.3.1 Goal

A requirement, as we have seen in the previous section, simply is a piece of text describing a system. It does not require any deep technical knowledge. It is therefore understandable by most *stakeholders* involved in the project. Stakeholders generally are software providers, users, customers, suppliers, etc. The main goal of a set of requirements is:

Requirements objective: to enable an agreed understanding between stakeholders.

Requirements are the first specification of the system. This is what should be understood under ”agreed understanding”. Requirements are the initial step in specifying what is expected from the software system.

2.3.2 Characteristics of individual requirements

To help reaching the above stated goal, each individual requirement needs to possess specific characteristics. Before going into more details, we can already state that any requirement shall:

- solve a stakeholder objective,
- be about the (software) system,
- be verifiable.

Requirements must be related to the needs of stakeholders. We should know why a given functionality of the system is needed. A mistake often made is to express requirements about users. Because requirements are used as specifications, we must have the possibility to actual check whether a system meets its requirements.

We here review the main characteristics provided to us by the 29148 standard (section 5.2.5) before discussing some examples.

- **Implementation Free.** Requirements shall specify what the system is expected to do without telling how the system shall do it. Requirements must be independent of implementation details, like choices of algorithms or architectures. For instance, ”The system shall sort packages” against ”The system shall sort packages using the QuickSort algorithm”. The issue is here why specifying an algorithm? A reason might be performance. In that case, the requirement must have a condition specifying the performance and leave it free how the performance is actually met. What if a more efficient algorithm (or platform etc ...) is found?

- **Unambiguous.** Requirements must be as simple as possible and have only one interpretation. This is very difficult to achieve because of the very nature of using English! By keeping each requirement short and following the syntax proposed by the standard, it is actually feasible to write clear requirements.

- **Singular.** To ensure clarity, a requirement shall include only one statement. If more is needed, write more requirements. This helps making requirements simple, easy to understand, and unambiguous.
- **Consistent.** Requirements must not contradict each other.
- **Traceable.** Requirements shall be upward traceable to stakeholders needs. We should keep in mind that the goal of requirements is to come to an agreement about what the system must do. A requirement shall therefore be justified by linking it to a stakeholder goal or need.

2.3.3 Attributes

Attributes are used to further explain and document a requirement. A requirement is the sentence following the proposed syntax. To review or analyse a requirement, it is important to know more about the context of this requirement. Here are some examples of attributes taken from the ISO 29148 standard:

- **Identification.** A requirement should be uniquely identified.
- **Dependency.** It is important to identify dependencies between requirements. It must be such that if a primary requirement is removed, the supporting requirement(s) can also be removed. Note when dealing with initial specification, requirements can remain quite abstract and dependencies should be avoided.
- **Source.** Source denotes here the originator of the requirement. Knowing the source – might be multiple sources – is important to identify which stakeholder must be consulted for clarification, modification, or deletion.
- **Rationale.** The rationale behind each requirement should be captured. The rationale shows the reasons why the requirement is required.

The standard proposes more examples. In practice, specific development teams or organisation may have their own attributes.

2.3.4 Characteristics of a set of requirements

A well-formed set of requirements shall possess the following characteristics:

- **Complete.** What is meant here by the standard is that the set of requirements contains everything pertinent to the system.
- **Consistent.** There should be no contradictory requirements.
- **Affordable.** The set of requirements is ”realistic” in the sense that it can be achieved under life cycle constraints (costs, schedule, etc ...)
- **Bounded.** It is important to stay within the user needs and not go beyond that.

2.3.5 Requirement language criteria

This is a very important point (See section 5.2.7 of the standard). The language used when writing requirements shall be as precise as possible. In particular, vague terms shall be avoided. Here is a non-exhaustive list taken from the standard of terms that SHALL BE AVOIDED:

- Superlatives: such as best, most.
- Subjective language: user friendly, easy to use, cost effective.
- Vague pronouns: it, this, that.
- Ambiguous adverbs and adjectives: almost always, significant, minimal.
- Open-ended, non-verifiable terms: provide support, but not limited to, as a minimum.
- Comparative phrases: better than, higher quality
- Loopholes: if possible, as appropriate, as applicable.
- Negative statements
- Passive voice: shall be able

2.4 Some remarks

In this section, we discuss some aspects of requirements.

2.4.1 From User Requirements to Low Level Requirements

At the very top level, requirements will specify the user needs. At that level of abstraction, requirements shall be understood by all stakeholders. The required technical background to read these requirements should be low. Later in the life cycle, requirements will be refined to software requirements, or sometimes even low level requirements. The goal of these refined requirements is to specify *to designers and programmers* what they need to implement. The level of detail is then much higher.

2.4.2 Requirements and AGILE

In modern development processes (like AGILE, SCRUM, XP, etc), the term ”feature” is often used. Nevertheless, the goal of a feature is very similar to a requirement: document a user need.

2.5 Conclusion

In this chapter, we presented a first specification method: requirements. Requirements are written in natural language using semi-formal syntax. They are the first specification model. They are very easy to use and are used much in practice.

The main drawback of requirements is that they do not show any structure. It is a very long list (realistic systems will have thousands of requirements) of text. It also does not tell about what interactions the system should have with users. This is the purpose of *use cases* – also called stories or scenarios – discussed in the next chapter.

2.6 Exercises

Exercise 2.6.1. Consider the requirements below. Are these requirements written according to the syntax described in the ISO standard? If not, rewrite the requirement according to the syntax.

- The system shall be able to process at least 40 executing jobs at a time.
- The system shall provide the means for the resource provider to see on which project this resource is working.
- If one of the resource disappears while it was performing a job, the system should re-queue the job.

Exercise 2.6.2. Requirements must have several characteristics, for instance, verifiable, consistent, ... Consider the following requirements. These requirements are not written according to the correct syntax. Analyse them according to the characteristics of well-formed requirements described in Section 2.3 and rewrite them according to the syntax proposed by the ISO standard.

- The information which is stored on the database can be accessed by any standard computer on the Company network.
- In order to obtain a Company car sticker the person must have a valid Company ID.
- The opening of the software shall take less than 3-4 seconds under normal working conditions.
- The user shall have access to French ? English dictionary (this is outside the scope of the application). The user shall ask questions or propose suggestions for words translations by mailing to the administrators.
- The software will be available 24hrs 365d/year.