Atom Free IT
for true business agility

Introduction requirements for architects

V4 22-11-2018

Index

1 Introduction

Although requirements are often defined by a requirements engineer or business analyst, the architect often makes a start. The architect focuses on the overall requirements that serve as input for the upcoming architectural decisions. Many projects and programs begin defining the architecture before defining the system requirements. This document is an introduction to requirements engineering for the architect. First, the various layers of requirements explained in section 2. Section 3 provides guidelines for drafting requirements.

2 Three requirement layers

We distinguish three main layers of requirements in a requirements specification: environment requirements, border requirements , and system requirements. This section explains these layers.

Requirements start at the primary stakeholders of the system. They have goals that they pursue in carrying out their duties. These goals are independent from the system, and thus contain no design choices for the system. These goals and resulting needs are recorded as environmental requirements. Environment requirements are often usable for more systems. The environment requirements are (partially) met by the system. This leads to the definition of border requirements, in which the system concept is applied to a specific stakeholder need. Border requirements are then translated into system requirements. System requirements describe the desired properties in terms of features or qualities of the system. Figure 1 shows the different concepts for requirements. The distinction between environment, border and system can also be found in other requirement approaches and architecting methods, but they often used other terms. For example: environment are called business requirements, border requirements are called user requirements, and system requirements are called functional requirements. That terminology is are useful for the functional/application perspective on the system, but our terminology can be used for all stakeholders.

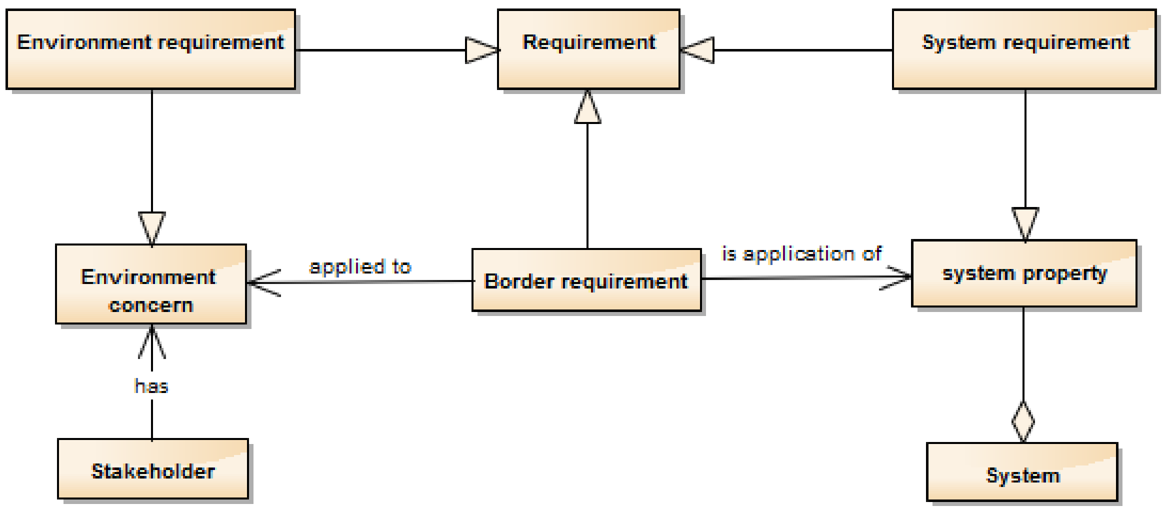

Figure 1 Three layers of requirements and their relations

Figure 2 shows the requirement layers of a software system. It shows that the environment requirements arise from the different (business) processes in the system context. From the (primary) business operation and its requirements, the usage requirements are derived. These are typically described in the form of system use cases. From the procurement and exploitation process follow the border requirements with respect to the design. These requirements cover, for example integrating the system with other systems or decomposing the system into parts. From the development and management process follow the border requirements that deal with the activities taking place to develop and manage the system. Think of the delivery of a release or testing of the system.

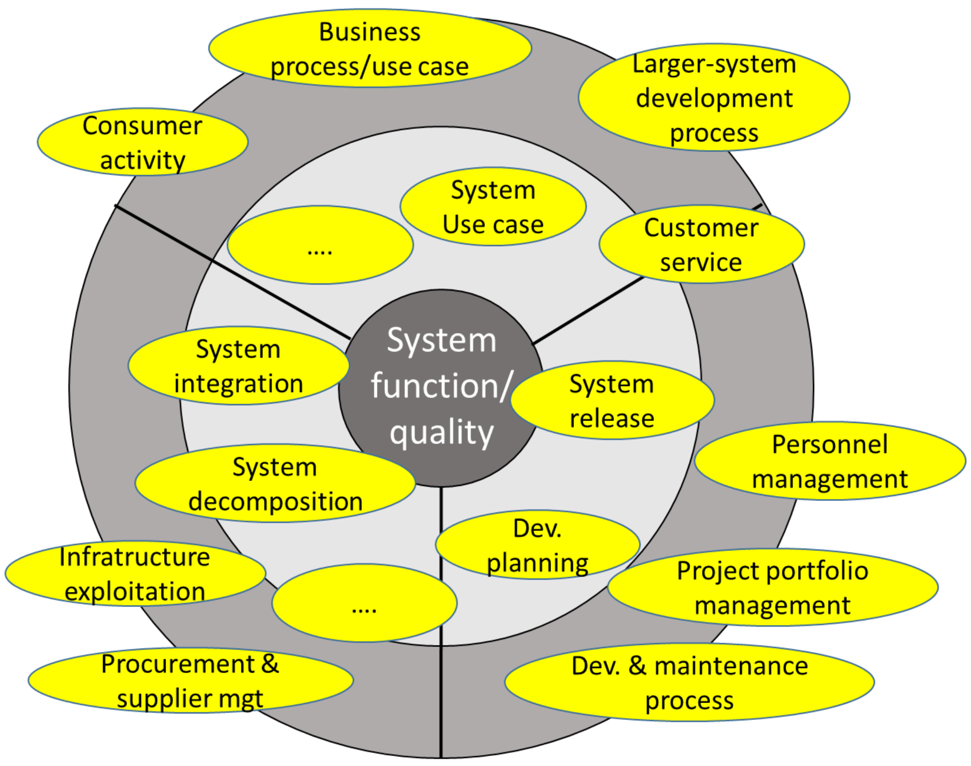

Figure 2 From context process to system property

2.1 Environment requirements

Environment requirements capture what stakeholders find important in the system environment. The focus is the stakeholder needs and business goals. The environment requirements are described independently of the solution (= the system). That is because the environment requirements exist, whether the system is made or not. Of course, it is not useful to consider various aspects that will never be of interest to the system or the system realization. On the other hand, it is not always clear in advance which stakeholder goals and resulting requirements are important for the system.

2.2 Border requirements

The border requirements say something about the relationship between the system and a specific stakeholder, or about the relationship with a process in the system context. These requirements are often described in use cases. The border requirements are not limited to use cases of the end user. They also include the use cases of the developer, such as linking components to an application, or making a technical design. The use cases of the application manager may also be relevant, such as installing an application on a server or rolling out a new version. Any border requirement should contribute to one or more environment requirements.

2.3 System requirements

The system requirements describe the desired properties of the system. This is in the form of a functional requirement, or a quality attribute requirement. Therefore, the system requirements are often called the functional specification (or even functional design). Each system requirement should contribute to one or more border requirements. A system function can for example be used in multiple use cases.

3 Guidelines for requirements

This document provides guidelines for drawing up requirements. There is much literature on requirements. For example, (Robertson and Robertson 1999) and (Gause and Weinberg 1989) address much more guidelines and aspects of working with requirements. The guidelines in this section provide a basis for the work of the architect. The following subsections provide guidance on individual requirements, a complete set of requirements and the writing style of requirements.

3.1 Guidelines for individual requirements

We present here some guidelines for drawing up a requirement:

- A requirement **fulfils a specific need**. There is at least one stakeholder who really needs the requirement, i.e., the owner of the requirement. If there is not such person, the requirement can be omitted. Otherwise following steps in the development might be disturbed by requirements that no actual stakeholder really cares about. Requirements that do not address a specific need are often statements of another kind:

  - Assumption: requirements that are in fact an assumption can take on a life of their own. This is often caused by a requirements author that writes down a requirement because he thinks a stakeholder needs it while he did not let the stakeholder validate it. Assumptions can be written down to stimulate stakeholders to think about their real requirements.
  - Design/implementation choice: requirements that are actually a design decision may push the solution too much in one direction. If there are alternative solutions, then the real requirement is often the WHY behind these solutions. A requirement that expresses a solution is only acceptable if there are no reasonable alternatives within the context of the project.
  - Situation descriptions: requirements that outline a situation without really stating a specific need or desires.
- A requirement is **verifiable**: it must be possible to check whether the system meets the requirement. An acceptance criterion might be used for requirements that cannot be tested within a reasonable time and effort. Verification does not imply that it is in the form of a software test. It could also be by inspection, analysis, or demonstration.
- A requirement is **realizable**: requirements that are not achievable within reasonable time and cost are nonsense to state. Be careful with omitting requirements to early though. This might cause that potentially good ideas are forgotten.
- A requirement is **atomic**: each requirement must display a single idea/concept that is indivisible. Multiple wishes in one requirement makes the requirement both difficult to plan and to implement. Especially when parts of such a requirement can be met and another part cannot. Indivisible does not mean that the requirement cannot be split. It may well be that the requirement can be detailed further. For example, the statement "The system must check the personal data" cannot be split. If it is known what the personal data comprises, then the requirement can be split into for example, "The system must check the name of the person" and "The system must check the address of the person".
- A requirement is **traceable**: it must be possible to trace a requirement back to its source. There are three direct sources for a possible requirement:
  - It is a requirement that comes directly from a stakeholder. So, the stakeholder is the source of the requirement, either stated verbally or in a document or other source of information.
  - It is a border requirement or system requirement, which is a realization of respectively an environment requirement or border requirement. The first requirement is a design decision for the realization of the second.
  - It is a subsystem (or component) requirement which is a translation of a system requirement as a result from the system decomposition in the system architecture.

  Besides traceability to the source of a requirement, other development artefacts, like designs, tests, and code should be traceable to the requirement that they cover.

3.2 Guidelines for a total set of requirements

The following guidelines will help in drawing up a whole set of requirements:

- Each requirement is **prioritized** within the total set of requirements: the importance of a requirement is always relative to any other requirement.
- Attach single requirement is only stated **once**. Duplicates cause disturbances in de development and often lead inconsistencies when the requirement changes.
- The set of requirements **sufficiently covers** the relevant domain. There are no aspects that are not covered in any requirement but that stakeholders have expectations about. There are no requirements implicit because some stakeholders find them too trivial to write down. Missing requirements can be found for example, by checking a standard list of quality attributes. Another starting point is the domain model to which the system applies. It is to be expected that there are requirements for each entity in the domain model. Notice that completeness is an illusion. You simply cannot know if you have all relevant wishes covered in a requirement. You can only try to avoid that you miss them.
- The set of requirements is **free of contradictions**. Requirements that contradict should be avoided. A system can simply never achieve two contradicting requirements. In the analysis phase and even the starting phase of the architecture contradicting requirements may exist.
- Requirements are **logically grouped**. Requirements dealing with the same things or functionally belong together, are managed together. A requirements management tool can help you create all sorts of views on a set of requirements.

3.3 Writing style guidelines

In case requirements are expressed in natural language the following guidelines are advised:

- For the **requirement nam**e the following applies:
    - A requirement indicating behavior, begins with the main verb, followed by the direct object, indirect object, possibly followed by, possibly followed by the main condition. For example:
        - Create termination letter to terminate policy.
        - Create policy for accepted request.
    - A requirement that defines a desired a state begins with the main noun (often a domain object), possibly followed by the state and condition. For example:
        - Policy number is always visible on the screen.
        - Company logo on every page.
- **Standard terminology** in the requirement description:
    - Requirements use the word "must" (English: must)
    - Assumptions use the word "will" (English: will)
    - Goals use the word "should". Specification of goals may only in environment requirements.
- **Avoid** the following words in the description:
    - Conjunctions: "and" indicates to 2 or more requirements, "or" indicates a more abstract requirement.
    - Generalizations such as "in general, usually, often, normal, typical, if possible, approximately."

  - "is, was, is going to do, ..." may be used in situation descriptions, but not in requirements.
  - Relative and subjective concepts such as "flexible, reliable, user-friendly, adaptable, fast, big, adequate, support, maximize, minimize, may, fault-tolerant, robust, efficient, ...". Such cases are often about a qualification of a quantity. But these terms are too subjective to be usable for verification.
  - Unclear quantities as "some, several, many, mostly, ...".
  - Suggestive terms such as "may, should, could, might, probably, perhaps, possibly, ..."
  - Explanatory terms such as "for example, such as, like, ...". This may be used for clarification in the notes, but not in the requirement itself.
- Create **understandable sentences**:
  - Avoid undefined jargon. Many people will not understand the requirement or worse, misunderstand it. Preferably, define the terminology in a domain model or a glossary.
  - Avoid negation sentences. It is better to specify what needs to be valid, then what must not be valid.
  - Avoid double negation. These are difficult to understand.
  - Avoid long sentences (over 15 words, many clauses).
  - Use the active form.
  - Make a complete sentence: subject, predicate, object, possibly indirect object or preposition object.

References

Gause, Donald, en Gerald Weinberg. *Exploring Requirements: Quality Before Design.* Dorset House Publishing, 1989.

Robertson, Suzanne, en James Robertson. *Mastering the Requirements Process.* Addison-Wesley, 1999.