1 Specification – some basic thoughts…

Now that we know how to find out 'the truth' about a needed solution using Mahan Khalsa's ORDER model (I personally call that phase the 'truth & trust phase'), we have to 'nail down our findings'. We need a methodology and tools to specify what we will do.

We need this for…

- estimating and simulating the effort, schedule, resource, risks, costs…
- writing a quote and offering our solution to our client
- writing a contract which represents the client's and our will
- writing test cases: at least in the end we must validate if the solution meets the client's needs and solves the pain or gain ☺ How could we do that if no criteria is defined? What should be tested and what should be fulfilled?
- …and many more things.

So how can we best specify an information technology or a software project?

1.1 (Software) Requirements Specification Basics

In IT, the requirements specification is the basis for several other project steps and documents. It is, for example, the starting point for the effort estimate, which leads to the cost estimate, which in turn is the basis for the quote. In addition, the specification is used for drawing up software contracts as well as developing and testing the finished product, which is developed during the project.

Here are a few definitions of terms used when talking about specifications in the IT sector:

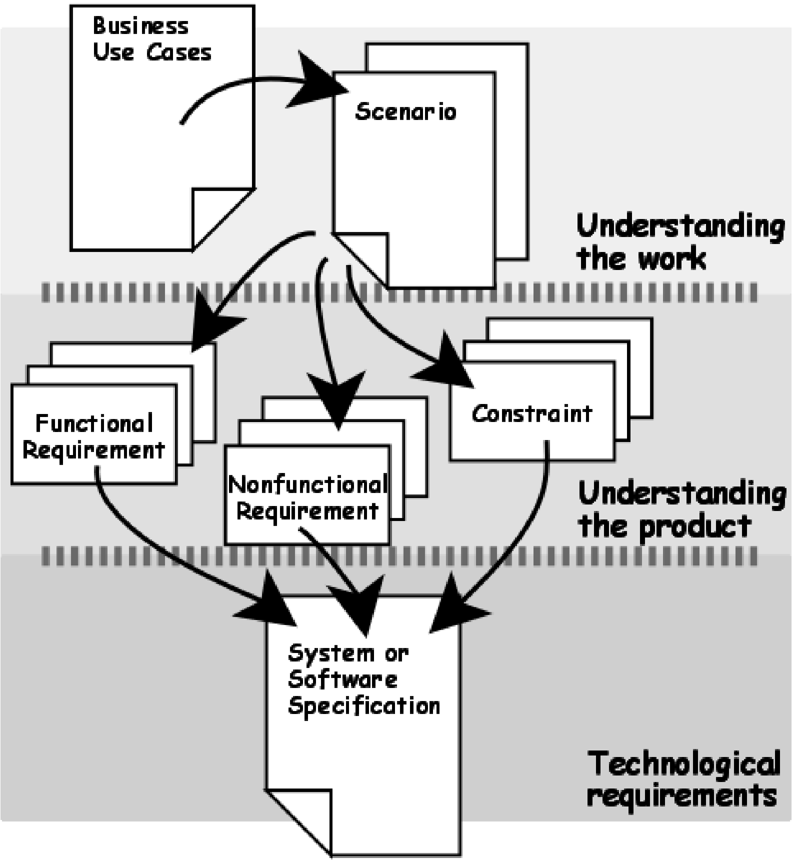
Product requirements document (PRD): In IT, a PRD is a general document, where the client (or their consultants) defines minimal requirements. A special form of the PRD is an invitation to tender ("Ausschreibung"), which is used to get comparable quotes for the essential requirements from different tenderers.

Functional specifications document (FSD): An FSD includes all rights and duties concerning the planned project with focus on the end product's specifications. It often contains legal aspects such as delivery and receipt conditions, prices, payment terms and conditions, liabilities and warranties.

Project contract, software contract: Software contracts are concluded for larger projects or for the sale of standard software. They usually only contain legal project and order conditions and refer to an FSD or a quote for more detailed definitions of requirements and services.

Quote: The quote is especially important for the client because it can be essential for the decision-making process. Quotes are a tightrope walk between objective and concrete work descriptions and sometimes contain some "hazy specifications" for marketing's sake. On the one hand we want to acquire a new customer, on the other hand we later need to implement the points we defined in the quote at the set conditions. A quote may also include a complete description of the product. In larger projects, however, the quote usually only contains a rough outline of the "work" to be delivered and refers to more detailed specifications (in the FSD, software requirements specification etc.). A quote includes prices, a rough or detailed project plan or schedule as well as general payment and delivery terms and conditions.

1.2 How to specify requirements


Business Use Cases
Scenario
Understanding the work
Functional Requirement
Nonfunctional Requirement
Constraint
Understanding the product
System or Software Specification
Technological requirements

Requirements development process in an IT project (Robertson, Robertson 2006, p.12)

When we start developing requirements, we first need to analyze the business for which the "work" will later be used. Scenarios (user stories, use cases) are created in cooperation with the client in order to define how they want their business cases to be processed. Then we need to identify detailed requirements from the point of view of an IT system (three different kinds of requirements).

In this phase, we should formulate our insights as technologically neutral as possible. The focus lies on the business process for which we are developing the IT solution which will support the client's business. Finally, the technological requirements are added and the final version of the requirements specifications is created as the basis for product development. Technical requirements also result from the client's requirements but are usually not easy to recognize and require our special attention during the specification process. (cf. Robertson, Robertson 2006, p. 11).

1.3 Types of requirements

1.3.1 Functional requirements

Functional requirements describe which functions the product must have to optimally fulfill the client's needs. It is important to know who will use the software to achieve which goals:

> *“A functional requirement is an action that the product must take if it is to be useful to its users. Functional requirements arise from the work that your stakeholders need to do. Almost every action – calculate, inspect, publish, or most other active verbs – can be a functional requirement.”* (Robertson, Robertson 2006, p. 9)
>
> Many misunderstandings and problems in projects arise from incomplete or vague functional requirements descriptions and usually lead to additional work and costs later on. Ambiguous terms and definitions are interpreted by developers so that they have the least development effort, even if this is not the functionality the client wants or needs. These mistakes are often realized very late and they usually entail high costs, e.g. because their correction requires several process steps (depending on the development method).

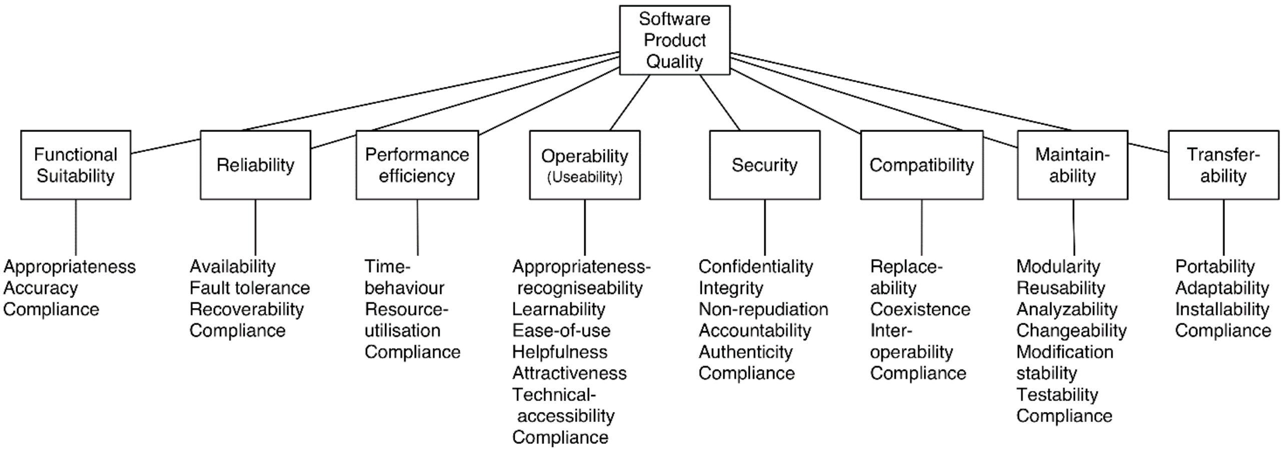
It is therefore essential that the functional product description is complete and consistent. All services and functionalities required by the user need to be defined.

1.3.2 Non-functional requirements

Non-functional requirements describe the ‘quality’ of an end product's features. The ISO/IEC 9126-1 norm distinguishes between six quality criteria for non-functional requirements: functionality, reliability, usability, efficiency, maintainability and portability, which are further divided into sub-categories.

Please note that the "functionality" characteristic here does not refer to functional requirements but to the degree of quality with which these functional requirements are fulfilled, i.e. the "functional quality". (This point has led to misunderstandings at exams in recent years, which is why I am emphasizing it now, hoping that it will answer any questions ☺)

Quality criteria according to ISO25010 (source: ISO document):



1.3.3 Constraints

Constraints (or framework conditions) are another set of requirements for product design or project implementation. An example for such a project condition might be: „*The product must be available before the beginning of the new financial year."* Failure to comply with this condition would render the whole product development useless.

In order to recognize constraints, we need to know a lot about the product's field of application and/or the client's business. This can lead to problems because project members sometimes think that the constraints are obvious (both on client and supplier side). They are especially hard to identify during the rough specification phase, so make sure to always ask the client during specification or quoting if special constraints apply.

1.4 Software Requirements Specification (SRS)

Various institutions have published templates and guidelines for building correct software requirements specifications. A very common (an kind of ‘historic’) one is the IEEE-recommended SRS method (830-1998):

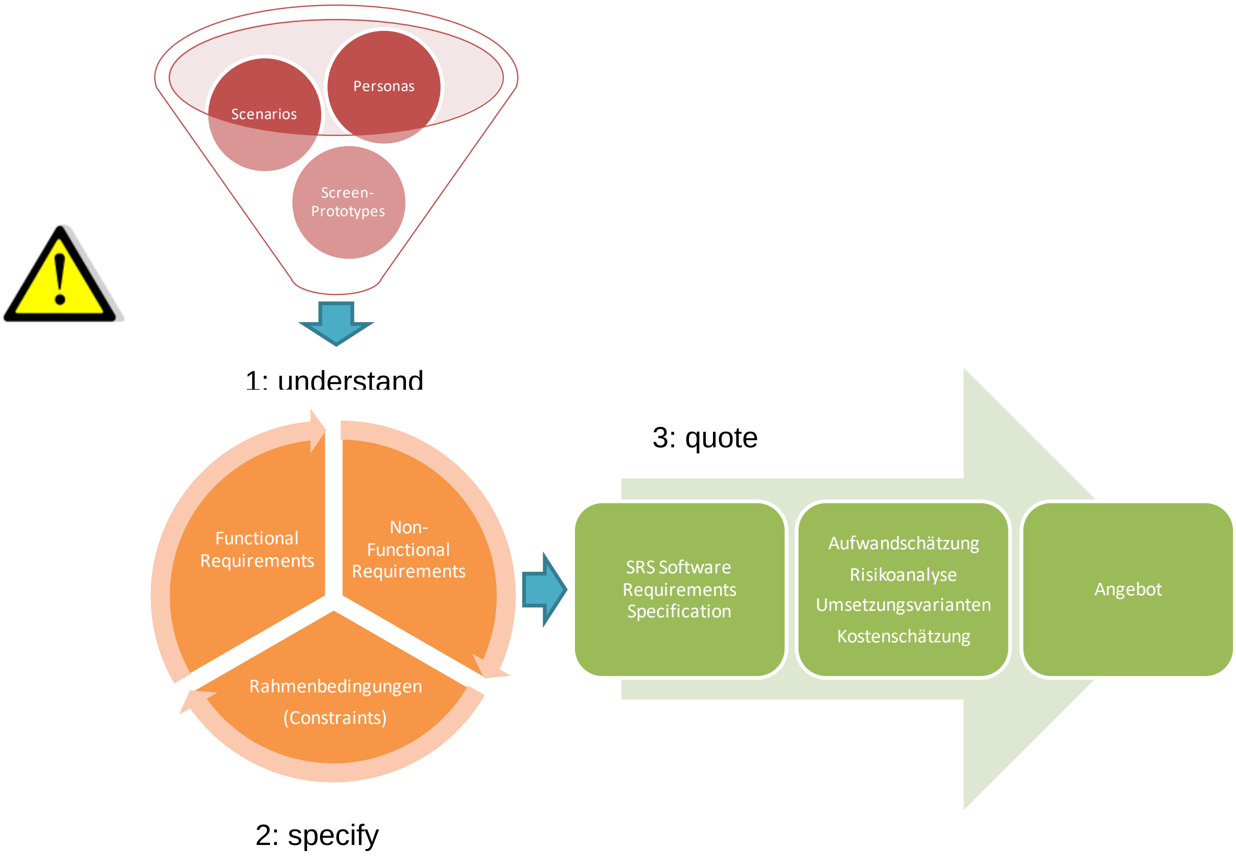
- Name of the software product
- Name of the producer
- Document's and/or software's version date

1. Introduction
    1. Purpose (of the document)
    2. Scope (of the software product)
    3. Definition of terms and/or abbreviations
    4. References to other resources or sources
    5. System overview (document structure)
2. Overall description (of the software product)
    1. Product perspective (compared to other software products)
    2. Product functions (summary and overview)
    3. User characteristics (information about intended users, e.g. educational background, experience, expertise)
    4. Constraints (for the developer)
    5. Assumptions and dependencies (characteristics that cannot be realized or have been postponed to later versions)
3. Specific requirements (in contrast to 2.)
    1. Functional requirements (depend on the type of software product)
    2. Non-functional requirements
    3. External interfaces
    4. Design constraints
    5. Performance requirements
    6. Quality requirements
    7. Other requirements

1.5 SRS Best Practices

The following specification process was developed in the author's company and has been applied ever since. It is introduced as an exemplary best practices approach for SRS processes in IT companies.

The funnel-shaped process for creating a quote based on a specification consists of three steps: 1. understand, 2. specify, 3. quote. See the following diagram:

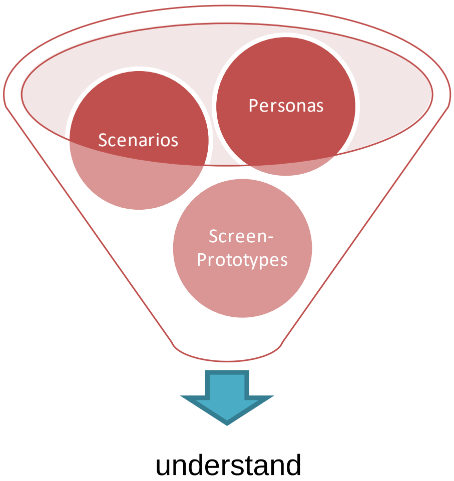


1.5.1 Understand

In the first step, we analyze the client's business in order to understand the client's processes and wishes needed for the creation of an optimal product or solution for their problem.

We need to ask questions, discuss the answers and communicate with the client. Under no circumstances should we advise the client on what they may need.

In the process above, information is collected in three different ways and then thrown into the big funnel from which the three requirements groups will be extracted:



Business analysis:

- Scenarios: business cases (user stories, use cases). In which scenarios will the future solution help the client or end users?
- Personas: What people make up the end user target group? Which roles need and want to/have to use the new solution?
- Screens / UI prototype / Mockups: Structure possible business cases by outlining them on the future user interface: usability engineering. How will these functions be executed? Screen layout, screen design etc. This may be omitted if no user interface exists (e.g. project for tuning some database queries or creation of a new printer driver ☺).

➔ **Goal:** To **understand** the **client** and their wishes and needs!

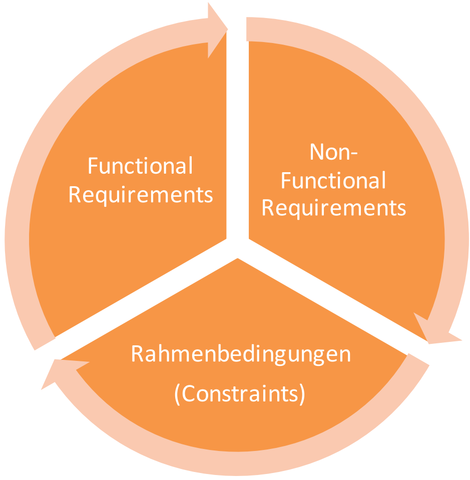
This phase often involves **workshops** with the client or the target group for which the solution will be developed.

Tip: The three elements within the funnel should be validated to check if the client's wishes were consistently and completely recorded. A simple trick is to see if something is "left over": There can be no scenario without a user (= actor), i.e. someone who wants to use this scenario (or an interface interacting with this scenario). For consistency's sake, we need to countercheck between scenarios and personas.

Remember: IT systems can also be actors. On the other hand, there can be no actors without a scenario. Their description within the system is either irrelevant or you forgot do define a scenario. Every scenario in which an actor interacts with the system must have a user interface (screen).

1.5.2 Specify and structure

In the next step, all information collected in the funnel is separated into three types of requirements:

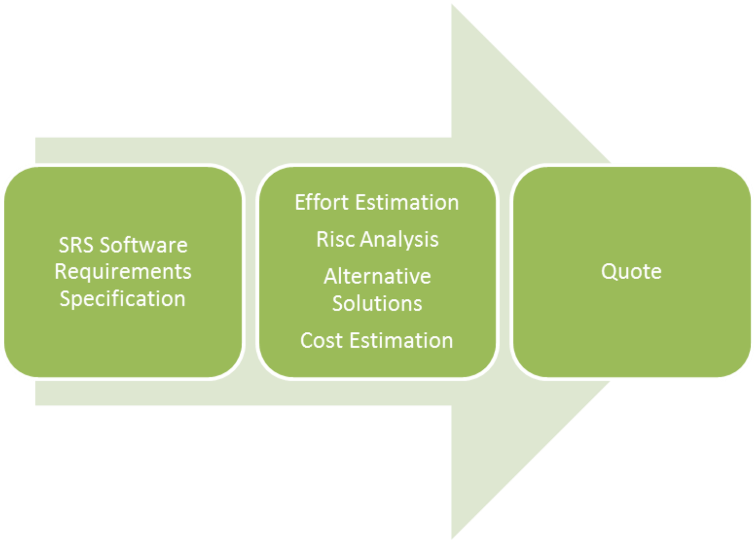


- Identify requirements **fully and consistently** and **write them down**.
- Make sure that they are **measurable** and **unique**.
- **All three requirements together** form the basis for an **SRS**.

The **SRS** is a written document (e.g. not a “Kaszettel” ☺) and refers to the three elements from the funnel, e.g. screens and scenarios are numbered so that **the SRS can refer to them at any time.**

1.5.3 Quote

After creating the SRS, a quote is drafted, based on the information gathered. The SRS’s level of detail at the time of quoting depends on the specific quoting process, as well as the pre-project phase. It can vary from “rough outline” to “ready for implementation”. In this best practices approach, the steps from SRS to quote are as follows:



1.6 Screen Prototypes, Scribbles and Mockups

Especially since the smartphones and tablets came up, usability and user-interface design became THE central topic when planning and developing a new software. Nowadays we sometimes do not even know or can’t influence on what type of client system the users will work with our software. Mobility is the buzzword of these days and so we have to deal with a huge variety of screens and devices where our software could be used on.

This is one of many reasons why the design and usability engineering of the user interface became a central topic during a software project’s specification phase. During the lecture units we will take some time for looking at typical mockups and screen prototypes. I strongly recommend that you use some kind of ‘optical / graphical description technique’ within your specifications whenever there is a user interface present in your software (you won’t need it when writing a SQL query or a printer driver of course ☺).

1.7 Progress monitoring

1. What is a product requirements document (PRD)?
2. What is a functional specifications document (FSD)?
3. What is a project contract or a software contract? Which types of software contracts do you know from your professional experience?
4. Name some important information contained in a quote. What is a quote based on? Which steps are needed before quoting (at least if you want to do it right ☺)?
5. What is an SRS? How is it structured (rough outline, several points)?
6. Outline a typical requirements development process.
7. Name the three types of requirements and briefly describe them.
8. Name a few key words from the six non-functional requirements groups according to ISO/IEC 9126-1.
9. Outline the presented best practices approach and roughly describe how it works.
    - Which possibilities exist to identify missing client information?
10. What are mockups and why and when is it a good idea to use these during a specification process?