1.6 The goal–design scale

Highlights

Goal-level requirement: why the customer wants to spend money on the product.
Domain-level requirement: support user tasks xx to yy.
Product-level requirement: a function to be provided by the product.
Design-level requirement: details of the product interface.

Tradition says that a requirement must specify

what the system should do

without specifying how.

The reason is that if you specify "how", you have entered the design phase and may have excluded possibilities that are better than those you thought of initially. In practice it is difficult to distinguish "what" from "how". The right choice depends on the individual situation.

We will illustrate the issue with an example from the Danish Shipyard (Chapter 11). The shipyard specializes in ship repairs. The values of orders range from $10,000 to $5 million, and as many as 300 workers may be involved in one order. Competition is extremely fierce and repair orders are usually negotiated and signed while the ship is at sea.

The management of the shipyard decided, for several reasons, to replace their old business application with a more modern one. One of their business goals was to achieve a better way of calculating costs.

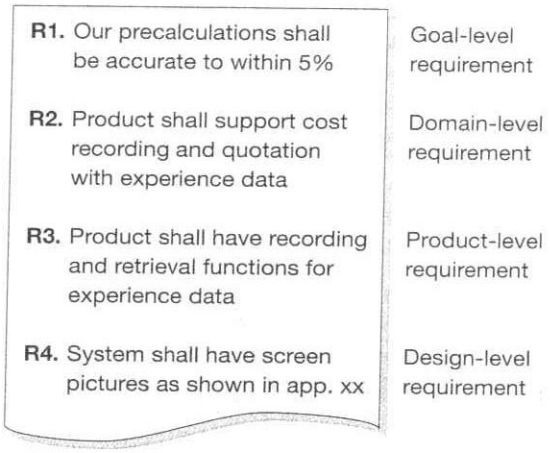
When preparing a quote, the sales staff precalculate the costs, but often the actual costs exceed the precalculation, causing the shipyard to lose money. Or the pre-calculated cost is unnecessarily high, causing the shipyard to lose the order. What is the solution to this? Maybe the new IT system could collect data from earlier orders and use it to support new cost calculations. Experience data could for instance include the average time it takes to weld a ton of iron, the average time it takes to paint 100 square meters of ship, etc.

Figure 1.6A shows four possibilities for the requirements in this case, which we will discuss one by one.

Goal-level requirement

R1 The product shall ensure that precalculations match actual costs within a standard deviation of 5%.

Fig 1.6A The goal-design scale



This requirement states the business goal, which is good because that is what the shipyard really want. Note that we call it a goal-level requirement because it is a business goal that can be verified, although only after some period of operation. Unfortunately, if you ask a software house to accept this requirement, they will refuse. They cannot take the responsibility for R1, because it requires much more than a new IT product: it is also necessary to train and motivate the shipyard staff, build up an experience database, etc., and even then it may be impossible to reach the goal. The customer has to take responsibility for that.

Domain-level requirement

R2 The product shall support the cost registration task including recording of experience data. It shall also support the quotation task with experience data.

This is a typical domain-level requirement. It outlines the tasks involved and requires support for these tasks. The analyst has carefully identified the right tasks. For instance, he hasn't specified a new user task to record experience data, because his knowledge of the shipyard and its day-to-day work tells him that then the recording would never be done. It must be done as part of something that is done already – recording the costs. Sections 3.6 and 3.8 explain more about domain-level requirements.

Could we give this requirement to a software house? That depends. If it is a software house that knows about shipyards or similar types of businesses, it may work. It doesn't matter whether the software house offers a COTS-based system with the necessary extensions, or whether they develop a system from scratch. However, if we choose a software house that is good at programming, but doesn't know about business applications, it would be highly risky, because they may come up with completely inadequate solutions.

Can we verify the requirement? Yes, even before the delivery time. We can try to carry out the tasks and see whether the system supports it. Deciding whether the support is adequate is a matter of assessing the quality. We discuss this in section 7.3.

What about validation? Can the customer reach his business goals? We can see that there is a requirement intended to support the goal, but we cannot be sure that it is sufficient. Here the customer runs a risk, but that is the kind of risk he should handle and be responsible for: he cannot transfer it to the software house.

Product-level requirement

R3 The product shall have a function for recording experience data and associated keywords. It shall have a function for retrieving the data based on keywords.

This is a typical product-level requirement, where we specify what comes in and goes out of the product. Essentially we just identify the function or feature without giving all the details. Section 3.4 tells more about this kind of requirement.

Could we give the requirement to a software house? Yes. If it is a software house that knows about shipyards there is no problem. Using COTS or developing from scratch are both acceptable. If we choose a software house that doesn't know about business applications, we would have to add some more detail about experience data, keywords, etc., then they should be able to provide the features we have asked for. Can we verify the requirement? Yes, before the delivery time. All that needs to be done is for us to check that the necessary screens are there and that they work.

What about validation? Here the customer runs the same risk as for R2. However, we run an additional risk. We cannot be sure that the solution adequately supports the tasks. Maybe the supplier has designed the solution in such a way that the user has to leave the cost registration screen, enter various codes once more, and then enter the experience data. A likely result would be that experience data isn't recorded.

Design-level requirement

R4 The product shall provide the screen pictures shown in app. xx. The menu points shall work as specified in yy.

This is a typical design-level requirement, where we specify one of the product interfaces in detail. Although a design-level requirement specifies the interface exactly, it doesn't show how to implement it inside the product.

R4 refers to the shipyard's own solution in app. xx. If they asked a business system supplier for R4, they might not get the best system. A supplier may have better solutions for experience data, but they are likely to use different screen pictures than those in app. xx. Insisting on the customer's own screen pictures might also be much more costly than using an off-the-shelf solution.

However, if the product was a rare type of system, the shipyard might have to use a software house without domain knowledge and have them develop the solution from scratch. In that case, R4 might be a very good requirement, assuming that the shipyard has designed the solution carefully. The shipyard would thus have full responsibility for ease of use, efficient task support, and its own business goals.

Choosing the right level

The conclusion of the analysis is: *choosing the right level on the goal-design scale is a matter of who you ask to do the job.*

You should not give the supplier more responsibility than he can handle. He may refuse to accept the added responsibility, or he may accept it but deliver an inadequate solution. Neither should you give him too few choices. It may make the solution too expensive, and if you haven't validated the requirements carefully, you may get an inferior solution.

In practice, the shipyard case is best handled through R2, the domain-level requirement. The main reason is that R2 ensures adequate task support and allows us to choose between many COTS suppliers. However, R1 is still important, although not as a requirement, but as a measurable goal stated in the introductory part of the spec. R4 may also be a good idea, not as a requirement, but as an example of what the customer has in mind. Of course, the customer shouldn't spend too much work on R4 since it is only an example.

R3 is rarely a good idea. The customer runs an unnecessary risk of inefficient task support and missed goals. Unfortunately, most requirements specs work on that level, and it is often a source of problems.

In the discussion above, we discarded R1, the goal-level requirement, because a software house couldn't take responsibility for it. Could we find a supplier that could accept this requirement? Maybe, but we would have to use a completely different type of supplier, for instance a management consultant such as PriceWaterhouseCoopers, Ernst & Young, etc. In their contract with the consultant, R1 would be the requirement, and R2 would be an example of a possible (partial) solution.

It is, however, likely that not even the consultant would accept R1 at a fixed price. Instead he might work on a time-and-material basis, tell the customer about other solutions and advise him whether experience has shown that 5% deviation was achievable in a shipyard, how to train staff, etc. In essence, the customer would get an organizational solution, possibly including some IT.

Quite often it is a good idea to use different requirement levels for different interfaces, or change from one level to another during the project. In section 1.7 and Chapter 5 we show ways of combining the levels.

What happened in the Danish Shipyard?

You may wonder what actually happened in the Danish Shipyard case discussed above. The full requirements specification is in Chapter 11. There are eight business goals, clearly stated in section 4 of the spec, which is an introductory section intended to help the reader. The real requirements are in section 5, which uses product-level requirements similar to R3 above. (Each sub-section has also an introduction that explains a bit about the tasks involved.)

When the system had been in operation for about a year, we assessed it and noticed that sales staff didn't use the experience data. Why? The financial manager, who had been closely involved in the development, said it was because sales staff were reluctant to use computers and they should be replaced by younger staff. A closer study revealed, however, that the system had no features to enable it to record experience data, and the feature for retrieving experience data was difficult to find and use from the quotation windows.

Further study revealed that the requirements didn't mention recording of experience data – it had accidentally been replaced by a reference to *retrieval* of experience data. The spec mentioned retrieval of experience data but didn't make it clear that this should be done during quotation. So, although all the requirements were verified at delivery time, nobody noticed that a highly important goal was lost.

How could this have been avoided? One possibility would be to trace the eight business goals to requirements and check them several times during development to ensure that the project remains on-track. A good quality manager would have ensured this. Another possibility would be to use domain-level requirements rather than product-level ones.

Asking "why"

In practice it is not always easy to choose the right level on the goal-design scale, and analysts often make mistakes. They should try to slide up and down the scale by asking "why" each requirement is necessary and "how" it can be achieved. Then they can select the requirement appropriate for the project.

Figure 1.6B shows some "why" questions for a device that measures neural signals in human patients. The customer developed medical equipment and had sub-contracted the software and part of the hardware development. One requirement said that the product should have a special mini-keyboard with start/stop button, repeat button, and a few other buttons.