Guide to Requirements SL-07

Template with Examples v2

Soren Lauesen 2011

Soren Lauesen
Guide to Requirements SL-07 v2 - Template with Examples
Version 2, 2011

Layout and cover: The author

Cover picture: Rob Gonsalves: "The Sun Sets Sail"
Kindly made available by Saper Galleries, East Lansing, Michigan, USA

The picture symbolizes the transition from
requirements (the bridge) to the product (the ship)

ISBN: 978-87-992344-1-7
Lauesen Publishing

Contents

Background

IT developers and consultants often ask for an exemplary requirements specification as a starting point for their specific project. This document is the guide to such a specification, *Requirements Template SL-07*. The specification itself is a template filled out with a complex example: *requirements for an Electronic Health Record system (EHR)*. A few points, for instance web access, had to be illustrated with examples from other areas.

You can download the requirements template from

http://www.itu.dk/people/slauesen/SorenReqs.html#SL-07

Requirements SL-07 uses a formulation based on experience with public IT tenders, in particular when the system is COTS based (Commercial Off-The-Shelf) so that large parts of it may exist already. Later, the approach proved advantageous for other kinds of acquisition too, and even for product development and agile in-house projects.

I wrote large parts of the template and the guide on request from the Danish Ministry of Research, Technology and Development, as part of their standard contract for software acquisition (K02). I am grateful to Bo Gad Køhlert, Anders Lisdorf and Garry Hopwood for careful reviews of the documents.

Earlier versions of the template have been used with success in 60 very different projects, tender processes as well as in-house projects, agile as well as waterfall, for instance: the new CMS of the Danish Defense, a pharmaceutical company's innovative document management system, a system for managing 1500 windmills.

Experiences from these 60 projects helped me write this version of the booklet - version 2. Among other things it includes a novel approach to integration requirements and systematic criteria for selecting a supplier.

I have learned that the approach is very powerful, but most practitioners get it all wrong the first time, particularly the tasks in Chapter C. After a bit of help they get it right. Half of them become great - and even improve the SL-07 approach.

Any comments - positive as well as negative - are most welcome and will help me improve future versions.

Soren Lauesen
The IT University of Copenhagen, May 22, 2011
slauesen@itu.dk
http://www.itu.dk/people/slauesen

1. The purpose of the template

IT requirements may be formulated in many ways. The main principle in Requirements template SL-07 is to strike a constructive balance between customer and supplier. They should for instance share the risk in a fair way. The customer should not write very detailed requirements, yet make sure that his real demands are met. And the supplier should have a chance to be innovative and build on what he has already.

The template achieves this by means of two columns for the requirements: Column 1 shows the customer's demands. Column 2 shows the corresponding solution imagined by the customer or proposed by the supplier (see examples on page 20). Depending on the kind of project, the parties can cooperate to improve the solution and/or modify the demands, or the customer can choose one of several suppliers according to the suitability of their solutions.

The experience is that column 1 (demands) is rather stable, while column 2 (solutions) changes as the parties learn about the possibilities. This also makes the approach suitable for agile development.

When customer and supplier are two different companies, there will usually be a contract in addition to the requirements. The requirements will be an appendix to the contract. There are no fixed rules for what to put in the contract and what to put in appendices.

Requirements template SL-07 uses an Electronic Health Record system (EHR) as the main example. The example is slightly simplified to make it easier to understand for readers outside the hospital area. The EHR area is very complex, so the example illustrates how to deal with difficult requirements. Only a few kinds of requirements had to be illustrated with examples outside EHR.

You can reuse large parts of the example in other projects. However, don't blindly reuse parts in [blue]. They are very EHR specific, or they are advice to the customer that isn't intended for the supplier.

1.1. Beware of template blindness

Using a template easily causes template blindness: Your worldview narrows down to what the template deals with.

It doesn't cover everything

The template doesn't cover all kinds of requirements for all projects, although it shows typical requirements within each requirement area. For your specific project, you must add the requirements needed in your case. Listen carefully to the customer and users and make sure their concerns are adequately covered by the requirements in one way or another.

It comprises too much

At the same time the template may comprise more than needed for your specific project. You easily include the unnecessary parts. The result may be that you pay far too much for the system, or that no supplier sends a proposal. As an example, the template contains requirements that will allow the customer to expand the system on his own. This may be costly and it is rarely necessary.

Look at each requirement and ask: What would happen if this was omitted? If in doubt, find out.

It includes very demanding requirements

A requirement may be relevant, but too demanding. As an example, the template requires response times around a second for systems that are used intensely on a daily basis. However, if the system is a website that users rarely access, response times may be longer without much harm.

1.2. The major requirements dangers

Experiences from tender processes show that some major problems occur over and over again. This guide can help you avoid the following dangers:

a. The requirements are on the wrong demand level. They may be so solution-focused that only a single supplier can meet them. Or they may be so business-focused that the supplier cannot take responsibility for them.

b. The requirements are too imprecise to verify. You cannot test whether they are met. Or they may be so open-ended that you cannot compare the supplier's proposals.

c. The requirements don't cover the important demands. Even if the requirements are met, the user needs and business goals are not met.

d. The major risks appear too late. Often much of the functionality is delivered early and the customer deploys part of the system. The hard parts are post-poned. Eventually it turns out that the supplier cannot deliver the hard parts, but due to time pressure, the customer ends up accepting the unsatisfactory system anyway.

We elaborate on these issues below.

1.3. The right requirement level

The requirements may describe the system in too much detail. The result may be that at most one supplier can meet them. On the other hand, requirements may be so high-level that the supplier cannot take responsibility for them. There has to be a balance. We distinguish between four requirement levels:

> Requirement 1 (goal-level: too business oriented). *The system must ensure that the number of medication errors is reduced from the present 10% to 2%.*

Comment: This requirement is on a too high level. It comprises business issues that are the customer's area of responsibility. The supplier cannot meet this requirement on his own. The customer is needed too, for instance to train staff and to record the necessary data.

> Requirement 2 (domain-level: adequate balance). *The system must support user tasks C1 to C7.*

Comment: The description of a *user task* explains what the two parties, user and computer, must do together. Task descriptions resemble "use cases" but don't specify what each party does. In a task you can also specify that something is a problem that should be eliminated. You don't have to specify how. This kind of

requirement allows the supplier to take responsibility for it, yet it can be met in several ways. The template uses this approach as far as possible.

> Requirement 3 (product-level: a required function with hidden purpose). *The system must show an overview of the patient's diagnoses.*

Comment: We cannot see the purpose of this overview. Is it to find a treatment, explain a new symptom, or write a patient report? As a result we cannot judge whether the supplier's solution is adequate. This is the traditional way of writing requirements and a major reason why customers don't get what they expect - although they get what they ask for.

> Requirement 4 (design-level: too solution oriented). *The system must show the patient's diagnoses as a hierarchical structure. Clicking on plus and minus must show the subordinate and superior diagnoses.*

Comment: This requirement describes a solution. It is inspired by a system the customer has seen. It specifies a particular solution, and suppliers with a different, but better solution must be discarded because it doesn't meet the requirement.

1.4. Precise (verifiable) requirements

The requirements must be so precise that they can be *verified*, i.e. we can decide whether the requirements are met. Precision has nothing to do with the demand level. As an example, requirements 3 and 4 above can be verified when the system is delivered. Requirement 1 can be verified when the system has been used for some time.

Requirement 2 can also be verified, but on a scale of degrees. Some systems may support the tasks well, others less well, but still adequately. The customer's staff can assess how well by walking through the tasks with each supplier, looking at the screens or screen outlines and noting down how well the tasks are supported (see Chapter 4). This assessment is essential for choosing the best supplier.

Here is a requirement that cannot be verified. It is not clear how to measure "easy to use" and decide when it is good enough:

> Requirement 5 (not verifiable). *The system must be easy to use.*

A requirement may be verifiable, yet express a demand so vaguely that we cannot compare the solutions. Here is an example:

> Requirement 6 (demand too open-ended: hard to compare the proposals). *The supplier is asked to describe his software integration strategy*.

Comment: This requirement can be verified already at proposal time. All you have to do is to check that the supplier has described a strategy. However it is hard to compare the strategies because they are "novels" in free style.

1.5. Cover the important demands

In practice we see many requirements that are precise, but formulated in such a way that we cannot judge how good the solution is. Requirement 3 above is an example. It asks for an overview of the patient's diagnoses, but we cannot see whether it supports the tasks where an overview is needed.

Requirement 2 is better because it specifies the tasks to be supported, and we can judge whether a given solution supports the tasks adequately.

The purpose of requirement 2 is to support some tasks. Another kind of purpose is to meet some business goals. For an EHR system, it might be to reduce the number of medication errors and reduce the length of hospital stays. In order to meet such goals, it is not sufficient just to acquire a new IT system. We may also have to change the business processes or carry out some existing tasks in a different way. Tasks may be supported better through IT, computerized entirely, or handled by someone else, for instance by the citizens or the suppliers.

Many projects have fine business goals, but nobody cared how to achieve these goals and how the new system should contribute. The result is usually that the expected results do not materialize. Section B1 of the template provides a simple way to trace business goals to requirements. Used properly it can help you rethink business goals, and come up with innovative solutions.

1.6. Early mitigation of major risks

The major technical risks in a project are usually response time with the full number of users, ease-of-use, and integration with existing systems. Deficiencies in these areas are virtually impossible to correct late in the project.

Section B2 of the template asks for an early proof of concept in order to mitigate the risks. Such a proof is expensive, however, so it isn't reasonable to ask the supplier to do it without a signed contract. However, he has to do it soon after. If he cannot provide an early proof, the customer may terminate the contract.

2. Gathering the requirements

The work of gathering and writing the requirements may seem overwhelming, particularly in a large organization. It is tempting to delegate the work to individual departments and let a central team edit the whole thing. Don't do that!

a. Each department will look at their own needs. They find it hard to look at it from a global company perspective. The result is that the requirements reflect the existing business processes without innovation and cross-departmental improvements.

b. The departments usually lack requirements expertise, and as a result the quality of the requirements becomes poor.

c. The central team doesn't obtain the necessary insight to understand the department, so they cannot improve the result - apart from language editing. One team expressed it in this way:

   We didn't understand what they wanted. So we just edited it into one big document and sent it to the potential suppliers. They should understand. We didn't realize until much later that the suppliers didn't understand it either. They just pretended so and told themselves: "we have to find out later".

2.1. Centralize the work

Let a small team carry out most of the work:

1. Gather demands, visions and wishes from the various stakeholders (including the departments, expert users, managers and clients).
2. Transform it into requirements according to this guide and the template.
3. Validate the requirements with the stakeholders and revise as needed.
4. Send the requirements to the potential suppliers, usually in cooperation with legal expertise.
5. Assess the received proposals in cooperation with the stakeholders.

The team should consist of 3-5 members with expertise from as many work areas as possible, including the IT function. At least one of the team members must have requirements expertise.

This approach can reduce the total work to one fifth of the decentralized approach. At the same time, the quality of the requirements increases dramatically.

2.2. Involve the stakeholders and maybe the suppliers

Although the team has broad expertise, it cannot know everything. Stakeholders must be involved too. Here are some ways to do it:

1. Interview users - expert users as well as ordinary users. Ask about present work, problems in the way things are done today, wishes and visions for the future.
2. Make the users show how they carry out their tasks today, in particular the rare, but difficult tasks.
3. Collect relevant documents, for example reports and forms used today, screen dumps, documentation of the existing database and the technical interfaces to the systems, statistics and operational reports.
4. Run workshops where stakeholders together with team members map the existing cross-departmental workflow and the ideal workflow.
5. Run brainstorm sessions or focus groups where participants inspire each other to new ways of doing things.
6. When new work processes are introduced, design them in some detail. As an example, when clients have to use electronic access rather than personal contact, customer staff has to work in a different way. This is often badly planned, but little is known about how to do it better. We suggest that you write task descriptions for these future processes and carry them out as role plays to check that the tasks are adequately described.
7. Visit potential suppliers. They often know how other customers utilize their products, and they can provide contact to them. They can also tell the customer about possibilities he didn't think of, or new ways to do things.

Some teams just list this very mixed information as requirements. Don't do that! It easily becomes a long wish list of requirements on a too solution-oriented level. Ask instead: Why is this wish interesting? When is it needed? What is the purpose? Which tasks would benefit? The result becomes broader demands that can be transformed into requirements.

2.3. Early change control

During the requirement process, you gather a lot of ideas, wishes, problems and potential requirements. Participants can spend oceans of time trying to agree on what to include, and this blocks progress. Instead park these issues in a safe place so that the team can progress. The "safe place" is a list of issues to deal with later - also called a list of *requests for change*.

Review the issues regularly and decide whether to transform them into requirements, into possible solutions, reject them, or keep them parked. You will often see that an issue that seemed impossible to deal with early in the project finds an easy answer later.

Continue the change control after signing the contract. You should observe that column 1 (the demands) are rather stable, while column 2 (the solutions) change as you learn about the possibilities.

3. Contract issues

When the system is developed in-house, there will rarely be a formal contract. The requirements specify what is to be delivered. Changes in requirements are discussed during development, and there are no financial penalties between the parties.

However, when customer and supplier are two different companies, there will usually be a contract *and* a requirements specification. The requirements specify what the supplier must deliver, and the contract specifies what to do when things don't proceed as expected. What to do when the supplier doesn't deliver on time or delivers a faulty product; or if the customer has forgotten an important requirement?

Lawyers specializing in IT contracts cleverly deal with all kinds of things that may happen during the project, in much the same way as programmers cleverly deal with all kinds of events that may happen in the system at run time.

Usually the requirements are one or more appendices to the contract. Other appendices may contain the supplier's description of the solution, prices for the deliverables, the implementation schedule, project management, testing, etc.

Requirements SL-07 uses a couple of principles that should be closely coordinated with the contract:

3.1. When solution doesn't meet demand

All requirements are written in tables. Column 1 specifies the customer's demands, for instance a particular task to be supported. Column 2 outlines example solutions, and later - in the contract - the supplier's solution (see example on page 20). The supplier will usually provide a more comprehensive description of his solution in a separate appendix.

Now what happens if at delivery time, it turns out that the supplier's solution doesn't meet the customer's demands? Who must pay for improving the solution? In many countries the default is that it is the customer's problem - he accepted the solution by signing the contract. In other countries, the rule is to protect the weak part - the party with the least understanding of the technicalities, in this case the customer.

Standard Danish contracts avoid the ambiguity by specifying that the customer's demands have priority. The supplier is responsible for meeting the customer's demands. He is responsible for the solution being adequate.

3.2. Rights to terminate the contract and try another supplier

Most requirements are low risk. If they have been "forgotten", they are easy to deal with late in the project. Others are high risk. They are so deeply rooted in the system architecture that they cannot be dealt with later.

To reduce the risk, Requirements SL-07 uses an early proof of concept (section B2). The customer - and maybe the supplier - has the right to terminate the contract in case the early proof of concept isn't satisfactory. This must be stated explicitly in the contract.

Customers are often reluctant to use this right and terminate the contract, even if the proof of concept shows that expectations are not met. The customer has already invested time and effort, and furthermore he would have to repeat the entire tender process. Make the pain less by stating in the tender announcement that proposals have to be valid for a period after the winner has been selected. Explain that this allows the customer to select the next best proposal in case the best doesn't meet the early proof of concept.

3.3. Exceeding expectations

Some requirements can be met to various degrees. Response times, for instance, can be longer than the customer's expectations, but still be acceptable. Requirements SL-07 suggests that the customer states his expectations and the supplier states what he offers.

If it is a tender process where the customer compares several suppliers, differences between expectation and proposal influence the supplier's scores. If the supplier proposes a longer response time, he will score lower on this point. What if he offers a shorter response time? Will he get an advantage? This has to be stated explicitly somewhere. The template states it in section A2.2: If the requirements say "or better", it is an advantage to exceed expectations.

3.4. Alternative solutions

A supplier may send a proposal with alternative solutions. This is useful if he can deliver an expensive solution that fully meets the customer's requirements, and alternatively a much cheaper solution that formally doesn't meet all the requirements, yet might be okay. He may offer alternatives for several requirements or requirement areas.

This puts a burden on the customer who has to assess all of this, maybe in different combinations. For this reason, some tender processes don't allow the supplier to specify alternatives. On the other hand it is risky. Page 20 (A2.3 - L2 in the template) shows a real-life example where the customer inadvertently would lose $3 million a year.

If the customer uses a selection approach with a modest number of sub-criteria (like section B3 and B4), it is rather easy to assess the marginal difference of two alternatives and the marginal effect on the final score. We suggest using the following approach:

1. For a set of alternative solutions, use the first one as the base. Assess the marginal effect on the final score for the alternative(s).
2. If there is a difference, choose the best alternative. If not, don't make a choice yet. It can be made after signing the contract.

3. Use the same approach for the other sets of alternatives. The result is one single score for the proposal.

In the A2.3 example, the result would be that the customer chooses the cheapest alternative, unless there was a significant business advantage of the expensive one.

In order for the approach to give reasonable results, the sets of alternatives must be independent of each other. The supplier should ensure this.

4. Assessing proposals

In public EU tenders the customer must assess the proposals on a numeric scale and choose the winner with the highest number. In many other cases it is also a good idea to assess on a numeric scale, even if it is not formally required.

The basic approach is that the customer looks at each requirement and assesses how well the solution meets it. The best is to get evidence for it, rather than opinions. Let the appropriate stakeholders participate in assessment of the various requirement areas.

As an example let us look at a requirement to support a specific task. Together with staff familiar with this task, carry out the task with the supplier's proposed system. Record how well the task is supported. You may try it on your own or - better - have the supplier show how the task would be carried out. If this is not possible because the necessary system parts don't exist yet, you must base the assessment on the supplier's screen outlines or other explanations of his solution. In this case, you might also note the risk of this not working in practice.

Based on the notes, you can give a single score for support of this task. Sections B3 and B4 of the template suggest scores on a scale of -2 (not supported or very badly supported) to 2 (very efficient).

For other types of requirements a similar approach should be used. For integration requirements, the supplier might show how existing integrations work, or explain how they will work. For documentation requirements, the customer can look at the supplier's existing documentation. For usability requirements, the customer can run usability tests or talk to existing users of a similar system that the supplier has delivered.

Sections B3 and B4 of the template suggest ways to combine the many scores into requirement areas, weigh them, include business goals and costs, and end up with a single score for the entire proposal. The sections also show how to guard against seemingly unimportant requirement areas being supported so badly, that the entire system may become a disaster.

5. Testing the system

Before the customer accepts the new system, he must test it - or have someone else test it. Otherwise, when defects are found later, he may lose his rights to terminate the contract or to request the supplier to repair the defects. In many countries the rule is that in order to win a court case, the customer must prove that reasonable tests wouldn't have found the defect at the time of delivery.

As a minimum the customer should verify all requirements (i.e. check that they are met). However, many errors don't relate to specific requirements but to the broad expectation that the system doesn't crash when users do strange things, or when the communication lines fail, etc. In order to test for this, we have to look at details beyond requirements. Here is a brief list of things to test for (see more in Patton, 2006).

1. Test that each requirement is met.
2. For each screen, test each button in various cases and test with boundary values and unacceptable values in each data field.
3. Test for exceptional events in the surroundings, for example loss of data communication and crash of external systems.
4. Verify that each branch in the program has been taken.

In medium-sized systems, thousands of test cases are needed and testing may take weeks. It is common to find hundreds of errors during testing. When the system is COTS-based (Commercial-Off-The-Shelf) large parts of it exist already. It is usually unnecessary to make detailed tests of these parts (i.e. points 2, 3 and 4 above).

Testing is often organized in stages:

Installation test: System delivery often starts with installation of the new hardware, software, etc. The purpose of the installation test is to ensure that the components work together and have basic functionality.

System test: The purpose of the system test is to check that requirements are met, screens work, etc. according to points 1-4 above. Special test data and database contents are used to allow testing all the cases.

Deployment test: The purpose of the deployment test is to check that the product can work satisfactorily in daily operation with production data.

Acceptance test: An acceptance test is a system test plus a deployment test. These two tests may be performed at different times or in combination.

Operational test: The purpose of the operational test is to check those requirements that can be verified only after a period of daily operation. It might be the response time under real load, breakdown frequency, task time for experienced users, qualifications of the supplier's hotline, etc.

6. Guide to the template sections

The rest of the guide comments the template, section by section. The gray text boxes are pieces of the template. Page 15, for instance, shows the front page of the template. Notice that the section numbers A, B . . . in the guide match the chapter numbers A, B . . . in the template.

You may freely download and use the template for a document as long as you clearly state the source and copyright notice, for instance as in the footer of the front page of the template.

Template chapters are numbered A, B, C rather than 1, 2, 3 . . . This is to avoid confusion with appendix numbers in the contract, which usually are 1, 2, 3. Appendix 2 might for instance be the requirements with the chapters A, B, C . . .

Be cautious about changing the chapter headings. Many people are familiar with the SL-07 structure and know by heart that Chapter C is tasks and Chapter H security.

The template starts with an introductory page to be deleted in your document. The next page is the front page of the final requirements (shown on page 15). It states the name of the system to be delivered. It is convenient to also define a short system name since several parts of the template refer to the system by name.

The front page also states the name of the customer, the name of the supplier, and a short description of what the delivery comprises. This helps the reader understand up front whether the delivery also comprises hardware, operation, etc. If the requirements specification is an appendix to a contract, the system name, customer name, etc. will be stated in the contract and are not needed on the requirements.

Some parts are blue and in brackets. These parts must be replaced with something else in the final requirements - or deleted. Other parts may often be reused.

The front page heading shows when the document was last changed and who changed it. These are document fields that MS-Word automatically updates when the document is printed or saved. The heading also shows the version number. Change the heading as needed to match your company standard.

The page after the front page is the change log. It shows what was changed when and by whom. Change it as needed to match your company standard.

Chapter A is background information about the project and a guide to the supplier on how to interpret the text and write a proposal. Chapter B explains the business goals for the project, what to prove early and how the customer selects the winner.

Chapters C to J specify what the supplier must provide on the day of delivery (i.e. at the end of acceptance testing). Chapter K specifies what the customer must provide. Chapter L specifies the supplier's responsibilities after the day of delivery.

Chapters K and L are often separate contract appendices and not requirements chapters. This is not important as long as they are somewhere.

Version 4.0 02-06-2011, 20:53
Last changed by: slauesen

Requirements specification for
[Electronic Health Record system
(below called the EHR system)]

Customer
[The … Hospital]
Supplier
…
The delivery comprises
[Software, operation and maintenance for an EHR system]

Contents

A. Background and supplier guide

A1. Background and vision

This section gives the reader a quick overview of the system and its purpose. Explain the main business goals (why the customer wants to spend money on the system), but don't go into detail (section B1 elaborates the business goals). Briefly explain the customer's present situation and his visions about the future.

Context diagrams for the present and future situations are good illustrations. The arrows show the flow of data. In surprisingly many requirements specifications, it is unclear what is to be delivered and who will do the integration with other systems. Make sure to show the system to be delivered as a single box with double-line borders. Show the integrations that the supplier must perform as double-line arrows.

In the example, the supplier must deliver his own system including or integrated with some medication system. He has to integrate with the existing NHC system and Labsys. The diagram shows that he is not required to integrate with new external systems. (As specified in section F10, a third party must be able to make these integrations.)

We often see customers writing a long story about their IT strategy, the historical development, etc. This is okay if it is limited to a few pages and helps the supplier understand the situation. However the story is often the customer's internal considerations or political statements that are not relevant to the supplier.

There may be a need for the customer - or his consultant - to explain the internal considerations in length, for instance the meetings held, the choices made, and the sources of the requirements, but do it in a separate paper. Not in the requirements.

Also make sure that the *background and vision* section doesn't contain requirements. Requirements have to be in boxes, as explained in the next section.

A. Background and supplier guide

A1. Background and vision

[Presently the customer has several old EHR systems that he wants to replace with one system to obtain:

1. more efficient support for the clinical work,
2. better possibilities for integration with future systems,
3. lower cost of operation.]

The customer expects that the supplier has a COTS system (Commercial-Off-The-Shelf system) that can meet many of the requirements. In return, the customer is willing to change his work processes to a reasonable extent, as long as the business goals are met (see section B1).

The present and future situations are illustrated with these context diagrams. The box with double-line border shows the system to be delivered. Double-line arrows show integrations that the supplier is expected to deliver. [There is presently insufficient integration between the old EHR system and the medication system. The customer wants an EHR system that includes a medication system.]

Figure 1: Existing system

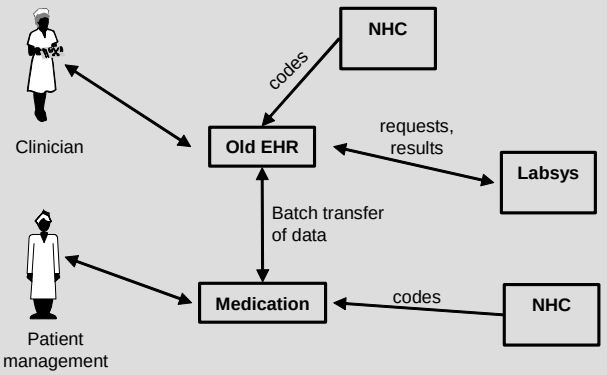
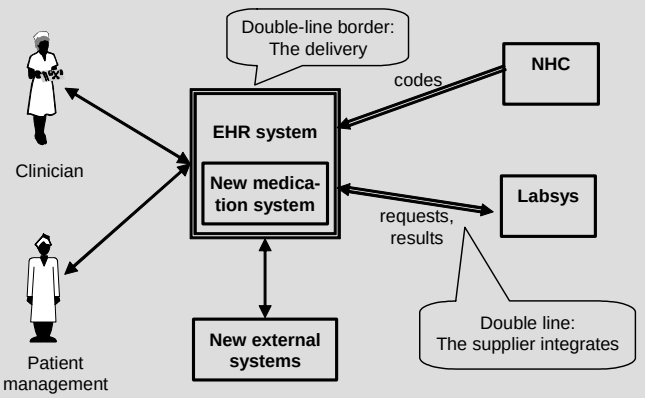


Figure 2: Vision for the new system



A2. Supplier guide

This section explains how the requirements are formulated and how the supplier's proposal is to be structured. Emphasize is on how to use the tables (the boxes), what are requirements and what are not.

The intent is that the supplier doesn't need other explanations than this section in the template. For instance he doesn't need to read this guide. For this reason there is a fictitious example of a general nature (a hotline application). In most projects you may use the example exactly as it is. Don't waste time constructing an example from your own project.

The example shows how the customer has specified variants and present problems to get rid of, and how he has described solution examples. The chapter also shows how the supplier specifies his solution.

A2. Supplier guide

This chapter explains the requirements format and how the supplier describes his solution.

All requirements are written in tables:
- Column 1 is the requirement (the customer's demand - what he wants the system to support).
- Column 2 may be a solution example or a proposed solution.
- Column 3 [may be the customer's rating, a reference to a sub-delivery, or something else.]

The requirements are organized in chapters according to their kind, e.g. Chapter C about user tasks to be supported, Chapter H about security. Within each chapter, the requirements are written in tables, e.g. a table with requirements relating to a specific task.

A2.1. Functional requirements - fictitious example

Functional requirements may be tasks to support, data to store, systems to integrate, etc. Here is a fictitious example without relation to the present delivery. The functional requirement is that the system must support a number of tasks, including C5, and preferably eliminate the current problems.

C5. Handle a request in hotline (fictitious example)
This task describes what a supporter does when he handles a request (a call).

Start: A user calls or sends an email; or the supporter has done something else and now looks for the next request to handle.
End: The supporter cannot do more about the request right now.
Frequency: In total around 500 calls per day. Per user: A maximum of 100 per day.
Users: First-line supporters with limited technical knowledge.

Subtasks and variants:	Example solutions:	Code:
1. Receive the request through phone or email. Or look at the pending requests.		
2. Record the request, particularly the user's phone, email and the cause of the request.	In case of an email request, the system automatically transfers data from the email.	
2p. **Problem**: Cumbersome to record, particularly when it is an on-the-spot solution.		
2a. It may be an update of an existing request. Find it.	The system shows possible matches with the caller's name or parts of it.	
3. Maybe transfer the request to 2nd line.		
…		

Some requirements are quality requirements where the supplier has to offer a response time, an availability percent, etc. In column 1 the customer explains what he needs in broad terms, e.g. high availability. In column 2 the supplier must specify the solution he proposes, for instance 98% availability.

The table lists the subtasks and problems of task C5. There is thus a requirement to support each of the table lines to some extent. The data about *Start, End* and *Frequency* are not in the table, meaning that they are not requirements (see more below).

The requirements are numbered. Variants of a requirement are marked with letters a, b, etc. In the example, the supporter may record the request (subtask 2) or find the request if it has been recorded already (variant 2a). Problems relating to a requirement are marked with the letters p, q, etc. A cross reference to a subtask, a variant, or a problem will look like this:
See C5-2 or *See problem C5-2p*.

Requirement. Column 1 of the table specifies the customer's demand, e.g. a subtask the system must support, or a problem it should eliminate.

Solution. Column 2 specifies the system's support of the demand. The column may show the customer's current imagination of a solution - if he has one. This is *not a requirement* or a wish, but only a possible solution to help the supplier understand the demand. In many cases the field will be empty. In the reply, the supplier will fill in the solution he proposes (see section A2.3).

Code. [Column 3 may be used in different ways depending on the nature of the project. The supplier may fill in column 3 with a code that specifies the sub-delivery (see section A2.4). Or the customer may specify priorities, or give a score for the supplier's solution.]

Text outside tables
In the example, *Start, End, Frequency*, and *Users* are outside the table. They are not requirements, but assumptions the supplier can make. In general, text outside the tables can serve several purposes:

A. **Assumptions** behind the requirements, for instance that the task must be supported for this kind of users, this frequency of use, etc.
B. **Requirement notes** that elaborate column 1 in the table. In principle they should be inside the table, but they don't fit well. One example is a list of access rights to the system.
C. **Solution notes** that elaborate column 2 in the tables. They are not requirements but example solutions. One example is various ways a user can look up a code in a table.
D. **Examples** and other information to help the reader understand the requirements.

A2.2. Quality requirements - fictitious example

Some requirements don't specify a function the system should carry out, but a quality such as an amount, a time limit or the like. Here is a fictitious example:

L2. Availability (fictitious example)
The system is out of operation when it doesn't support some of the users as usual. The cause of the breakdown may be: …

Availability requirements:	Example solutions:	Code:
1. In the period from 8:00 to 17:00 on weekdays, the system must have high availability.	In these periods the availability is ____%. (The customer expects 99.5%).	
2. In other periods …	…	

The customer still states his requirement in column 1, but now column 2 specifies how he wants to measure or structure the reply. In addition, he may state what he expects, i.e. what is sufficient. The customer may accept something less than expected, but will then give the solution a lower score on this point. If the supplier offers more than 99.5% it will not give him an advantage. The customer may specify that more *is* an advantage by writing 99.5% **or better**.

A2.3. The supplier's reply

In the reply, the supplier fills in column 2 to specify the solution he proposes. He may show alternative solutions. Here is a possible reply to C5 above:

C5. Handle a request (fictitious example)

…

Subtasks and variants:	Proposed solutions:	Code:
1. Receive the request through phone or email. Or look at the pending requests.		
2. Record the request, particularly the user's phone, email and the cause of the request.	~~In case of an email request, the system automatically transfers data from the email~~. (The system has a semi-automatic capture of email. The user must initiate the recording.)	
2p. **Problem**: Cumbersome to record, particularly when it is an on-the-spot solution.	A. The present version records the caller based on the email. B. Release 18 will provide buttons for easy recording of the most frequent causes.	
2a. It may be an update of an existing request. Find it.	The system shows possible matches with the caller's name or parts of it. The system also provides phonetic search. See screen 12 in App. x.	
3. Maybe transfer the request to 2nd line.	See screen 13 in App. x.	
…		

The supplier has changed the heading of column 2 from *example solution* to *proposed solution.* He has crossed out the example solution that isn't relevant anymore (C5-2). He has shown two solutions to C5-2p, the present one and the one in the next release. He has shown a solution to C5-2a that exceeds the customer's example. For C5-3 he refers to a longer description of the solution.

Here is a possible reply to the quality requirement above:

L2. Availability (fictitious example)

The system is out of operation when it doesn't support some of the users as usual. The cause of the breakdown may be: …

Availability requirements:	Example solutions:	Code:
1. In the period from 8:00 to 17:00 on weekdays, the system must have high availability.	A: In these periods the availability is 99.8%. (The customer expects 99.5%). B: In these periods the availability is 99%. Will reduce yearly operational cost by roughly $3 million (see …)	
2. In other periods …	…	

The supplier has specified two solutions: Solution A exceeds the customer's expectation. Since the customer didn't ask for 99.5% **or better**, the supplier doesn't get an advantage by exceeding the expectations. Solution B has a slightly lower availability, but reduces the yearly operational cost significantly.

However, it is hard for the supplier to specify a figure if he doesn't know what the customer expects. Should he offer an expensive system with high availability or a cheaper, but less available one? For this reason the customer should state what he expects, for instance 99.5%.

If the supplier proposes a lower availability, he will not be excluded but he gets a lower score. If the customer cannot live with a lower availability, he must write the required percentage in column 1. If the supplier offers a better availability, he may or may not get an advantage. This has to be explained somewhere. The template states it in section A2.2: If the requirements say "or better", it is an advantage to exceed expectations:

Expects 99.5%:	Higher is not an advantage.
Expects 99.5% or better:	Higher *is* an advantage.

When the customer writes his expectations, the supplier gets more freedom, but the principle should not be abused. The customer's expectations should be realistic and important. As an example it is unrealistic and unimportant to expect a response time of 1 second for a complex report that is rarely used. A serious supplier may take great effort to offer an expensive system to match the expectations, while a cheaper one would suffice.

[In some tender processes the supplier is not allowed to specify alternatives. This is very risky. For the availability requirement above, the customer may not be aware of the high cost of asking for the last 0.5%. The result may be that suppliers don't propose the cheap solution, although it probably would suffice. See more in the guide, section 3.4.]

A2.4. Codes

[Column 3 may be used in different ways depending on the nature of the project. The customer may specify priorities, or give a score for the supplier's solution.

Another possibility is that the supplier fills in column 3 with a code that specifies the delivery:

1	Part of a COTS system.
2.x	An extension of a COTS system, but the extension is covered by the ordinary maintenance agreement. Will be available from delivery stage x.
3.x	Custom-made software or an extension of a COTS system that is *not* covered by the ordinary maintenance agreement. Will be available from delivery stage x.
4.y	Part of a future release that will be supplied under the ordinary maintenance agreement. It will be available from release y.
5	No support is offered for this requirement.
Alt.z	Alternative solutions are offered. This solution is part of alternative z. Use alternatives sparingly. See the guide, section 3.4.]

A2.5. Practicalities about formatting

The template is an MS-Word document. It uses heading styles on level 1, 2 and sometimes 3, plus a special heading style, *TOC without number.* The headings automatically generate the table of contents. In order to improve the overview, some headings have a forced page break. It may be changed through Format ? Paragraph ? Line and Page Breaks.

Tables use the table style *Requirement Table.* It has borders of 3/4 point. It has top and bottom cell margins of 0.5 mm. Column 1 uses *Column1* style (Ctrl+1). It has a hanging indent of 0.75 cm. Within a table cell, you tabulate with Ctrl+Tab, since Tab alone moves the cursor to the next cell.

B. High-level demands

B1. Business goals

This section of the template contains the business goals of the system, arranged in a table to show how the goals are to be met. Column 1 is the goal; column 2 the vision - the solution in broad terms; column 3 the requirements that make it possible. It is emphasized that the goals aren't requirements to the supplier, but background information. Column 4 allows the customer to state the deadline for meeting the goal. When stated, it is the deadline for the joint effort of the supplier and customer. The supplier should bear in mind that the customer also needs time for the organizational implementation.

The business goals serve several purposes:

a. They tell the supplier what the customer wants to achieve.
b. They are important criteria for choosing a solution.
c. They help the customer check that the crucial requirements are included.

In the example, goal 1 (efficient support of all user tasks) is a very broad goal that depends on a lot of requirements. It is stated in such a way that it allows the customer to discard solutions that poorly support one or more tasks. As an example, the surgeon needs a good overview of the patient's situation in order to make the right decision. It must be possible to discard a system with a poor overview screen although this is just one of 1000 details in the system. Sections B3 and B4 explain how this can become part of the selection criteria.

In the example, the customer had identified goal 3, *continuous improvement of the work processes*. However, he hadn't realized that this required a new organizational structure - an *advisory board* - that should develop, test, and deploy new standard procedures for patient treatment. This required IT support, but the customer didn't realize it until the goal table had to be filled in.

Don't specify a lot of goals. If there are more than 10, check that they are not just requirements. We often see "goals" of this kind: *It must be easy to print consumption reports*. Although this was important to one of the stakeholders, it is a simple system requirement, not a business goal. A business goal is about the results of the entire organization, not just something the computer can do.

If you cannot write something reasonable in column 2, it may be a sign that the goal is not a true business goal, but a requirement. As an example, if the goal is: *It must be easy to print consumption reports,* it will be hard to write a large scale solution. If you cannot write a true business goal, simply leave column 2 blank.

Measuring the goals: A really good goal can be measured and compared against the existing state of affairs. Goal 2 is clearly of this kind. Goal 1 can be measured on a subjective scale of degrees (e.g. 1 to 5), or as the number of tasks performed per person per day. Goal 4 could be measured as operational costs before and after system deployment. Although the goals can be measured, the customer may not want to reveal the measurements. They might tell the supplier which price the customer is willing to pay. Section B4 gives an example of how to avoid it.

B. High-level demands

This chapter explains how the customer's business goals are met through the requirements and how to mitigate high-risk requirements.

B1. Business goals

The customer's reason to acquire the system is to reach some business goals. The customer expects that the system contributes to the goals as stated below. The supplier can rarely reach the goals alone. Customer contribution is needed too. This means that the goals are **not requirements** to the supplier. They are shown in a table only to provide overview.

All goals are important and the sooner they can be met, the better. Some goals are crucial to meet at a specific date, for instance for business or legal reasons. Such deadlines are shown in the table.

Goals for the new system	Solution vision	Related requirements	Deadline
1. Efficient support of all user tasks.	[All relevant data are available during the task without switching between systems. All parties can see the health record.]	Support for all tasks in Chapter C. System integration, particularly F2. Adequate response times in L1.	
2. [Reduce medication errors from 10% to 2%.]	[Avoid manual steps - record the prescription immediately. The system checks for validity, drug interaction, etc.]	[Support for task C10 (clinical session), in particular subtask 2 (assess the state of the patient). Support for task C11 (prescription), almost all the subtasks.]	
3. [Continuous improvement of the work processes.]	[Easy to set up and modify standard treatment plans. Easy to integrate the system with new lab systems, etc.]	[Support for task C30 (advisory board). Requirements in sections E4 and F10 (system expansion and integration with new systems).]	
4. Lower operational costs.	Replace several expertise-demanding systems with one.	Support for all tasks from the previously separate work areas.	
5. [Meet the new EU rules on ...]	…	…	[1-1- 2012]

B2. Early proof of concept

This section lists certain high-risk aspects of the project - things that cannot be amended late in the project. To reduce the risk, the supplier has to provide an early proof that it is possible to deliver what is required.

Most of the functional requirements are low-risk. It is for instance straightforward to add some fields and tables to the database, or some simple screens to the user interface. Most high-risk areas concern the quality requirements. In general, *quality is not an add-on feature*.

The template mentions that the contract allows both parties to terminate the contract if the early proof fails. Make sure this is the case. See more in section 3.2.

Requirements B2-1 to B2-5 specify what is to be tested early. Column 2 provides an example of how to test it. The supplier may change it to his own test proposal. He also specifies when the proof will be ready. (Sometimes a supplier may even have a proof before the contract is signed.)

In general these tests may be expensive, so it is not reasonable that the supplier has to carry them out before signing the contract.

B3. and B4. Selection criteria

In a tender process, the customer chooses the supplier according to some selection criteria. To ensure a fair process, the selection criteria must be objective and known to the supplier in advance. Usually only the following major criteria are relevant:

1. The total business value of the solution.
2. The risk to the customer.
3. The delivery time.
4. The total cost to the customer.

How can these criteria be combined? A pragmatic approach that works well with a small number of proposals is to look at them in light of the actual case, discard proposals that are clearly inferior to the rest, and come up with arguments that can point out the winner among the rest.

However, according to for instance the EU tender rules, this is not allowed because the customer can fiddle with the criteria so that his favorite supplier becomes the winner. You have to define the criteria and their weights up front so that you end up with one single number for each proposal.

This causes problems that often force a customer to select the "wrong" supplier.

Problem A: The customer may be forced to select a supplier that is too weak in an important business area *X*, because this area didn't show sufficiently in the criteria. The supplier was very strong in most areas and even though he scored very low for X, he came out a winner.

Problem B: The business value depends on only some of the requirements. The non-business requirements are not expected to contribute to the value. The customer is forced to select the proposal with the highest business value, but the solution may be very inconvenient in a seemingly unimportant area.

B2. Early proof of concept

Some requirements are high-risk and the supplier may not be able to deliver what he promised in his proposal. If this is detected late in the project, the customer may terminate the contract, but this is a disaster to both parties. Usually the customer chooses to accept the inadequate system, possibly with compensation from the supplier. To reduce the risk, the customer requires an early proof of concept for the high-risk requirements.

[According to the contract, both parties can terminate the contract if the early proof fails.]

The following requirements are considered high-risk. In his reply, the supplier must state how he will carry out the proof of concept and when. The date must be stated as the number of workdays after signing the contract. The customer expects 40 workdays or less.

High-risk areas where an early proof of concept is required:	**Example of proof:**	**Code:**
1. [Efficient support of clinical sessions (task C10)].	A prototype of the necessary computer screens (maybe a paper mockup) is assessed by expert users. Can be done within __ workdays.	
2. Usability (all requirements in section I1).	A prototype (maybe a paper mockup) is usability tested with ordinary users. Can be done within __ workdays.	
3. Response times with the required number of users (all requirements in section L1).	A test setup is used to simulate the required number of users. The response times are measured. Can be done within __ workdays.	
4. Possibility for third-party expansion of the system (sections E4 and F10).	An independent software house studies documentation of parts of the system and the technical interfaces in order to assess whether it is adequate for expanding the system. Can be done within __ workdays.	
5. Integration with other systems.	A test setup which demonstrates the data exchange. Can be done within __ workdays.	

An example could be unacceptable security features. Although security is not a business goal, the business value would be reduced dramatically.

The template shows two alternative solutions to all of these problems: selection criteria B3 and B4. B3 expresses the business value in dollars; B4 expresses the business value in score points. (Delete either B3 or B4 in your document.)

Traditionally, selection criteria are part of the tender material, but not of the requirements. SL-07 includes them in the requirements in order to show how they can be expressed and how they link to requirements.

Minimum criteria

B3 as well as B4 use minimum criteria to deal with problems A and B. In the request for proposal, the customer has specified a minimum score for each requirement area. Later the customer gives each proposal a score for each area.

The template uses these scores: -2 (not supported or very inconvenient), -1 (inconvenient), 0 (as today or just sufficient), 1 (efficient), 2 (very efficient).

Here are the reasons behind the minimum scores in the example:

Area		Minimum scores
C1	Admit patient before arrival. Support is not really needed for this task. The customer can just keep his existing admission system.	-2
C10	Perform clinical session. This will get a weight through business goal 2, but to prevent problem A, we demand that the system supports clinical sessions at least as well as the present one.	0
C11-C…	Medication (considered one area). This too must be supported at least as well as today to prevent problem A.	0
…	…	…
D	Data to record. We don't assess this separately. It is done indirectly when we assess how well the system supports the tasks.	N/A
…	…	…
F10	Integration with new external systems. We want this to be better than today. We assess it when we select the winner and re-assess it during the early proof-of-concept.	1
H1	Login and access rights for users. This must be at least as good as today to protect against problem B.	0
H2-5	Other security (one area). We accept that it is a bit worse than today.	-1
I	Usability and design. This must be at least as good as today. We assess it when we select the winner and re-assess it during the early proof-of-concept.	0
J2	User training. This must be at least as good as today. It also gets a weight because it is part of the investment costs.	0
J4	Data conversion. This just has to be sufficient. It is a one-time issue.	0
L1.	Response times. This must be at least as good as today. We assess it when we select the winner and re-assess it during the early proof-of-concept.	0
…	…	…

The principle of giving a score for each requirement area rather than each requirement is important. A requirement specification may contain more than thousand requirements and giving a score for each of them is overwhelming. Furthermore, it is virtually impossible to assess a requirement in isolation. They interact.

Dividing into requirement areas as in the example brings us down to around 40 areas - a more manageable number. Furthermore it is more meaningful to assess support of a task or another group of related requirements.

In general there may be several requirement areas where the customer can accept a proposal that is worse than today. It would be foolish to reject an otherwise great proposal because it is weak in a few areas. However, it shouldn't be weak in too many areas. The example deals with this by means of another minimum criterion: There may not be more than 3 weak areas.

B3. Selection criteria [Alternative 1: Highest benefit in dollars]

[This section is important for public tenders. The suppliers must know the selection criteria and their weights before writing a proposal. In commercial acquisitions, the customer need not state any criteria.]

The customer discards proposals that don't meet certain minimum criteria. Among the rest, the customer chooses the proposal with the highest net benefit measured in $.

Scores: The customer gives each proposal scores for the requirement areas shown in the table below. To ease later comparison, all tables have space for several proposals (columns A, B and C). The scores are measured on this scale: -2 (not supported or very inconvenient), -1 (inconvenient), 0 (as today or just sufficient), 1 (efficient), 2 (very efficient).

Minimum score: It may happen that a proposal has a high net benefit, yet is too inconvenient in a seemingly unimportant area. The minimum scores guard against this. In the table each requirement area has a minimum score. A proposal that scores lower than the minimum in any of these areas will be rejected.

To illustrate the approach the table shows fictitious scores for proposal A. Notice that it is acceptable that a proposal scores -2 or -1 in some areas. The customer doesn't want to reject an otherwise good proposal because it is weak in a few areas. However it shouldn't be weak in too many areas. For this reason there is also a limit on the number of scores below zero (last line of the table).

Requirement area	**Minimum score**	**Score**		
		A	**B**	**C**
[C1. Admit patient before arrival.	-2	1		
…				
C10. Perform clinical session.	0	1		
C11-C… Medication (considered one area).]	0	2		
…				
D. Data to record. Assessed through the task support.	N/A	N/A		
…				
F10. Integration with new external systems. Also assessed during early proof of concept.	1	1		
H1. Login and access rights for users.	0	0		
H2-H5. Other security (one area).	-1	-1		
I. Usability and design. Also assessed during early proof.	0	1		
J2. User training. Also included in investment costs.	0	0		
J4. Data conversion.	0	1		
L1. Response times. Also assessed during early proof.	0	0		
…				
Number of scores below 0.	max 3	1		

Method B3: Benefit in dollar

Method B3 computes the financial value for each business goal. The customer has computed the potential value of the goal per year. As an example, efficient support of the tasks might save each employee an hour a day. For 7000 employees, this means saving around 1000 employees, or 40 million $ a year.

Fraction obtained: A proposal may have weaknesses that will reduce the actual benefit to a fraction of the potential. For each proposal and each business goal, the customer estimates this fraction. As an example, if the proposed system can save only 0.5 hours a day, the fraction is 0.5. In principle the fraction may be higher than 1. This happens if the proposal exceeds the customer's expectations.

Risk: A proposal may be risky, for instance because the solution hasn't been tried somewhere else, or the solution is very sketchy, or the supplier needs a long early-proof-of-concept to test it. For each proposal and each business goal, the customer estimates the risk that the benefit will not materialize.

Based on the potential value and the proposal-specific fractions and risks, the total yearly benefit is computed for each proposal.

Total cost: The cost in the example consists of the product cost as offered by the supplier, the cost of hardware and other equipment that the customer has to buy, the cost of training the staff, and the operating costs for a period of 5 years.

Notice that all of these may differ between proposals. Some proposals need more customer hardware than others; some need more staff training than others, etc.

Net benefit: The net benefit - the bottom line - consists of the total benefit for 5 years minus the total cost for 5 years. However, the benefit will not materialize during the period when the product is delivered and deployed. This "lost benefit" is subtracted as a "cost".

Method B3 will now force the customer to reject proposals that don't meet the minimum criteria, and among the rest select the proposal with the highest net benefit.

Benefit

The benefit of the proposal is based on a financial value for each business goal. To illustrate the approach the table shows fictitious figures for proposal A.

Potential: The customer estimates the potential benefit of the goal per year. Measured in million $.

Fraction: For each proposal the customer estimates the fraction of the potential benefit that this proposal can reach if delivered as promised. It is stated as a number with one decimal, normally in the range from 0.0 to 1.0. Example: The potential cost saving of efficient task support is estimated to one hour per day per employee. A proposal that might save only half an hour will have fraction = 0.5.

Risk: For each proposal the customer estimates the risk that the fraction will not be realized as promised. The risk is estimated based on how detailed the solution is, whether the relevant part of the solution exists, whether it has been tried elsewhere, the supplier's domain knowledge, and the time proposed for the proof of concept. Example: The supplier has sketched a detailed solution but it doesn't exist yet. The risk is estimated to 30%.

Value: Computed as Potential * Fraction * (1-Risk)

Business goal	**Poten-tial**	**Fraction**			**Risk**			**Value**		
		A	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
[1. Efficient support of clinical tasks	40	0.5			30%			14		
2. Reduce medication errors	10	1.0			10%			9		
3. Continuous improvement	10	1.0			40%			6		
4. Lower operational costs (included in the costs below)										
Total benefit per year (million $)]	**60**							**29**		

For each proposal the customer estimates the net benefit. First the total benefit is computed for a period of 5 years. Then the costs of deploying and operating the system are subtracted. The result is the net benefit for 5 years. Notice that all the figures may vary between proposals.

Benefit for 5 years	**A**	**B**	**C**
Total benefit for 5 years	**145.0**		
Product cost	20.0		
Customer hardware costs	10.0		
Staff training	5.6		
Operating costs for 5 years	20.0		
Lost benefit during delivery	22.0		
Total costs for 5 years	**77.6**		
Net benefit for 5 years	**67.4**		

Method B4: Benefit in score points

Method B4 uses the same minimum criteria as B3, but it doesn't calculate the benefit in dollars but as a weighted sum of score points.

Total score: In the example, we have added a column to the list of requirement areas: a weight for each area. We can now calculate the sum of weighted scores for each proposal. This is the total score points.

Weights: How do we determine the weights? One possibility is to give each area a priority, for instance between 1 and 5. The priority is now the weight. However, this is hard to justify from a business point of view. Furthermore it can be extremely hard to make stakeholders agree on one area being priority 1 and another priority 5.

We suggest that you try to find weights that reflect the impact of the area, for instance the number of staff affected or the effect on quality. In the example we have used the business value of the area, but disguised it as a weight. Notice that many areas have weight zero. Better support of them has little impact - as long as the minimum score is met.

Total cost: The cost in method B4 is computed exactly as for method B3.

Bottom line: In B3 we subtracted cost from benefit to get the net benefit. Then we selected the winner according to the highest net benefit. We cannot do this in B4. It doesn't make sense to subtract cost in dollars from benefits in score points.

However, it makes sense to divide the two. This gives us the number of score points per million dollars. Method B4 will reject the proposals that don't meet the minimum criteria, and among the rest select the one with highest score per million dollars.

Comparison: The main advantage of B4 is that we don't have to estimate the business value in dollars. Even if we could, we might not want to reveal it to the suppliers. It also allows us to put weight on quality aspects that cannot be estimated in dollars. Finally, the whole procedure is somewhat simpler.

B4. Selection criteria [Alternative 2: Most bang for the buck]

[With this alternative the customer doesn't have to specify the benefit in $. And he doesn't have to reveal to the supplier how much he expects to gain. Risks and delivery time are not included below.]

The customer discards proposals that don't meet certain minimum criteria. Among the rest, the customer chooses the proposal with the highest score per $.

Scores: The customer gives each proposal scores for the requirement areas shown in the table below. To ease later comparison, all tables have space for several proposals. The scores are measured on this scale: -2 (not supported or very inconvenient), -1 (inconvenient), 0 (as today or just sufficient), 1 (efficient), 2 (very efficient).

Weight: Each requirement area has a weight that reflects the impact of the area, for instance the number of staff affected or the impact on the customer's service quality.

Minimum score: It may happen that a proposal has a high net benefit, yet is too inconvenient in a seemingly unimportant area. The minimum scores guard against this. In the table each requirement area has a minimum score. A proposal that scores lower in any of these areas will be rejected.

To illustrate the approach the table shows fictitious scores for proposal A. Notice that it is acceptable that a proposal scores -2 or -1 in some areas. The customer doesn't want to reject an otherwise good proposal because it is weak in a few areas. However it shouldn't be weak in too many areas. For this reason there is also a limit on the number of scores below zero (last line of the table).

Requirement area	**Weight**	**Minimum score**	**Score**		
			A	**B**	**C**
[C1. Admit patient before arrival.	5	-2	1		
…					
C10. Perform clinical session.	40	0	1		
C11-C… Medication (considered one area).]	10	0	2		
…					
D. Data to record (assessed through task support).	N/A	N/A	N/A		
…					
F10. Integration with new external systems. Also assessed during early proof of concept.	10	1	1		
H1. Login and access rights for users.	0	0	0		
H2-H5. Other security (one area).	0	-1	-1		
I. Usability and design. Also assessed during early proof of concept.	5	0	1		
J2. User training. Also included in investment costs.	0	0	0		
J4. Data conversion.	0	0	1		
L1. Response times. Also assessed during early proof of concept.	5	0	0		
…					
Number of scores below zero		max 3	1		

For each proposal the customer computes the total weighted score and the costs of deploying and operating the system for a period of 5 years. Finally the score per million $ is computed.

Score per million $	**Total A**	**Total B**	**Total C**
Total score: weighted total of the scores	**80**		
Product cost	20.0		
Customer hardware costs	10.0		
Staff training	5.6		
Operating costs for 5 years	20.0		
Total costs for 5 years	**55.6**		
Score per million $	**1.4**		

Variations

There are many variations on the B3-B4 themes above. In B3 we could select the winner according to the financial benefit per invested dollar. This corresponds to the managerial situation where we have a limited amount of money to invest and choose the projects that give the largest return on investment.

We could also be more precise and calculate the internal rate-of-return (IRR), taking into account the varying benefits and costs over a period of years.

We might add maximum criteria on the cost, e.g. our budget doesn't allow us to invest more than 10 million dollars. And add minimum criteria on the benefit, e.g. we won't invest in something unless we get at least a 20% return on investment.

For B4 we could include the risk of not getting the full score points, and subtract "missing" score points for the period where the system is delivered.

More importantly, we can vary the score scale, for instance from (-2, -1, 0, 1, 2) to (1, 2, 3, 4, 5). This will make the B4-selection more sensitive to cost differences and less to quality differences. In general it is a good idea to test the weights and scales by imagining hypothetical proposals with different scores and costs, and check that the selection criteria make sense.

Finally, you should remember that there is a high level of uncertainty and risk in large IT projects. Fiddling with details in the calculations will have little impact compared to these risks. Fortunately the selection of a winner is often *robust*: Even if we vary the weights and estimates quite a lot, the same winner comes out.

(Intentionally left blank)

C. Tasks to support

This chapter describes the user tasks to be supported. A user task is something user and computer do together from start to end without essential interruptions. A good start point is something that happens in the user's world, for instance that a client calls. A good end point is that nothing more can be done about the case right now - the user deserves a "coffee break" (*task closure*).

The first task in the chapter is C1. It starts when the secretary receives a message about a patient.

The task ends when the patient has been admitted, put on the waiting list, or the call has been parked - waiting for additional information. The table lists the subtasks involved. As far as possible, the user decides which subtasks to carry out and in which sequence.

Subtask 1 records the patient. We don't specify whether user or computer does it. Initially we don't know how much the computer is doing; it depends on the supplier's solution. Good support is that the computer does most of it, for instance copies the patient data automatically when the message is electronic.

Not use cases: Tasks resemble "use cases" but don't describe the interaction between user and computer. The task only describes what they do together. In a task you can also specify that something is a **problem** that should be eliminated. You don't have to specify how. In the example it is a problem that some messages don't use the standard MedCom format correctly.

We explain more on tasks versus use cases on the next pages.

What are the requirements? Right after the chapter heading, the template states that the requirement is to support subtasks and eliminate the problems as far as possible. This means that column 1 of the table contains the requirements. Column 2 is example solutions, but not requirements. Things outside the table are assumptions under which the requirements must be met, or information to help the reader understand the rest.

Work areas

In order to assess how well a task is supported, we have to know what kind of users we deal with, the environment where the task is carried out, etc. We might specify this for each task, but often several tasks are carried out by the same kind of users in the same environment. It is convenient to bundle the tasks accordingly. Such a bundle is called a *work area*.

In the template we describe each work area as an introduction to the bundle of tasks. We describe the user profiles (roles) and maybe the environment. The user profiles explain the user's IT experience, domain experience, motivation, etc. Some users may work in several work areas, possibly with different roles in different areas.

C. Tasks to support

The system must support all user tasks in this chapter, including all subtasks and variants, and mitigate the problems. The subtasks are numbered for reference purposes. They don't have to be carried out in this sequence, and many of them are optional. The user decides what to do and in which sequence. A subtask may also be repeated during the same task.

Work area 1: [Patient management]

[This work area comprises calling in patients, monitoring waiting lists …]

User profile: [**Doctor's secretaries**. Most of them are experienced IT users with good domain knowledge. They communicate well with medical staff.]
User profile: [**Clerical staff** …]
Environment: [Office …]

C1. [Admit patient before arrival]

[This task creates an admission record or continues the admission process if it has been parked. Most admissions can be recorded in one task. The rest have to be parked, e.g. because some information is missing. It is important that the system ensures that parked admissions are not forgotten (see task …)]

Start: [Message from medical practitioner, from another hospital … The message may also carry missing data or be a reminder about a parked admission.]
End: [When the patient has been admitted or recorded on the waiting list, or when the admission has been parked while the missing data is on its way.]
Frequency: [In total: Around 600 admissions per day. Per user: A maximum of 40 per day.]
Difficult: (never)
Users: [Initially a doctor's secretary, but the case may be transferred to someone else.]

Subtasks and variants:	Example solutions:	Code:
1. [Record the patient. (See data description D5).]		
1a. [The patient is in the system. Update data.]		
2. [Admit also a healthy companion.]		
3. [Record the admission, including the initial diagnosis. (See data description D1 and D6).]		
3a. [Transfer data from medical practitioner, etc.]	[The system uses the MedCom protocol.]	
3p. [Problem: The electronic messages use non-standard codes and formats.]	[The system allows editing of the transferred message.]	
3q. [Problem: The patient may have several admissions at the same time at different hospitals and departments. It is hard to see who is responsible for nursing and where the bed is.]		
4. [Find a meeting time for the patient and send an admission letter.]	[The system supports printed letters as well as electronic, confidential letters.]	
4a. [Put the patient on the waiting list.]		
4b. Essential data is missing. Park the case with time monitoring.		
4c. Transfer the case to someone else, possibly with time monitoring.		
4d. Maybe reject the case.		
5. [Request an interpreter for the meeting time.]		

C2. [Admit immediately]

[This task handles patients who arrive in an emergency without notice …]

C1. Admit patient before arrival - a simple task

In a task description we use imperative language (*admit patient*) in order to hide who does what. The full task description consists of these parts:

ID and name: Tasks are numbered C1, C2, etc. To avoid too much renumbering during requirements elicitation, we bundle the tasks and start each bundle with a round number. In the example, C10 is the first task in the next bundle (the work area *patient treatment*).

Introduction: A short introduction to what the task is about.

Start and end: A task should be something that is carried out from start to end without essential interruptions. Notice that a task may start for more than one reason and end in more than one way.

The start signal (the **trigger**) should be something that happens in the user's world outside the computer. One example is that a patient takes his medicine. This triggers a task where nurse and computer together record the event. Avoid computer-focused triggers such as *the user wants to record a medicine intake*. This doesn't help the supplier understand what caused this wish and whether the system might support it better, for instance through barcode scans of the medicine package and of a special bracelet attached to the patient.

Task C1 may be parked because some data are missing (subtask 4b). Although the task isn't completed in a logical sense, it is completed task-wise for now. The user starts doing something else. This pattern is very common and it is important that the system supports it well, for instance through warnings about overdue, parked tasks.

Frequency: The task frequency for the entire organization and for the user. The frequency for the entire organization helps the supplier estimate the necessary computer capacity. The frequency for the user indicates the importance of an efficient user interface. However, since these figures are outside the table, they are not requirements. The real requirements are response times (L1) and usability (I1).

Difficult: Situations where the task is particularly difficult to carry out, for instance because it is done under stress or requires high precision. Note that task C1 has no difficult situations while task C10 has one.

You cannot readily observe difficult situations but have to ask users about them. Use *difficult* sparingly. It is outside the table and thus not a requirement. Early in the requirement process you may write *difficult*, but try to remove it later. Often we can describe it as a *problem* with one of the subtasks. Then it is easy to check whether the supplier has a good solution. We can also describe a difficult situation as a *separate task*. This helps us check that the supplier supports it well.

Users: The users who carry out the task, the environment, etc. Omit this information if the work area specifies it.

Subtasks and variants: Column 1 is a list of subtasks, variants and problems. The subtasks are numbered sequentially. This is for reference only. Subtasks are basically optional (need not be carried out) and may be carried out in many sequences. The user decides what to do and in which sequence.

C1. [Admit patient before arrival]

[This task creates an admission record or continues the admission process if it has been parked. Most admissions can be recorded in one task. The rest have to be parked, e.g. because some information is missing. It is important that the system ensures that parked admissions are not forgotten (see task …)]

Start: [Message from medical practitioner, from another hospital … The message may also carry missing data or be a reminder about a parked admission.]
End: [When the patient has been admitted or recorded on the waiting list, or when the admission has been parked while the missing data is on its way.]
Frequency: [In total: Around 600 admissions per day. Per user: A maximum of 40 per day.]
Difficult: (never)
Users: [Initially a doctor's secretary, but the case may be transferred to someone else.]

Subtasks and variants:		Example solutions:	Code:
1.	[Record the patient. (See data description D5).]		
1a.	[The patient is in the system. Update data.]		
2.	[Admit also a healthy companion.]		
3.	[Record the admission, including the initial diagnosis. (See data description D1 and D6).]		
3a.	[Transfer data from medical practitioner, etc.]	[The system uses the MedCom protocol.]	
3p.	[Problem: The electronic messages use non-standard codes and formats.]	[The system allows editing of the transferred message.]	
3q.	[Problem: The patient may have several admissions at the same time at different hospitals and departments. It is hard to see who is responsible for nursing and where the bed is.]		
4.	[Find a meeting time for the patient and send an admission letter.]	[The system supports printed letters as well as electronic, confidential letters.]	
4a.	[Put the patient on the waiting list.]		
4b.	Essential data is missing. Park the case with time monitoring.		
4c.	Transfer the case to someone else, possibly with time monitoring.		
4d.	Maybe reject the case.		
5.	[Request an interpreter for the meeting time.]		

C2. [Admit immediately]

[This task handles patients who arrive in an emergency without notice …]

Notice that we use imperative language also for subtasks (record the patient). You may write several lines to describe a subtask or a problem. C1-3q and C10-2 are good examples. If you need more space, a requirement note below the table is better.

Variants of a subtask are indicated by letters a, b, etc. A variant means that the subtask may be carried out in more than one way. As an example, we may either record the patient (subtask 1) or find the patient in the system (1a). Problems relating to the subtask are indicated by letters p, q, etc.

Many subtasks consist of recording or using data, but some subtasks comprise more, for instance advising other people (subtask 5), dispensing medicine, sending a letter. It is important to include this even if it is done manually today. The supplier may have a solution that the customer hasn't imagined.

Problem = current problem: Column 1 also lists problems. A problem must be something that troubles the user in the present way of doing things. Problem 3q is a good example. The customer wants the supplier to eliminate the problem. We often see analysts stating an imagined future problem, for instance that it will be difficult to provide overview of the data. This is not the intention with "problem". If you want to mention such issues, do it in column 2, which deals with the future.

Solutions: The customer may write example solutions in column 2. Later the supplier writes his proposed solution here (see section A2.3 of the template).

As a customer, write sample solutions sparingly. Don't force yourself to write something "clever" here. Only write something if it is a non-trivial solution. Be explicit in column 2 about who does what, e.g. *The system shows* or *The user selects*. Avoid passive statements such as *The results are recorded*.

C2. Admit immediately

This task handles patients who arrive in an emergency without notice. Although the task resembles C1 there are differences, and C2 may need different support.

Don't worry about the same subtasks appearing in several tasks. We need to check the support of them in the different contexts. A programmer will try to reuse code - great, but the analyst doesn't program. The analyst should ensure that all use contexts are supported properly.

C10. Perform clinical session - a complex task

The most important activity in a hospital is examining and treating the patient. How many tasks are involved? Is *examination* one task and *treatment* another task? If we study what actually goes on, examination, treatment, and other activities are often carried out within the same short period of time. It is important that the computer supports this mix well.

So the task starts when the clinician starts dealing with the patient and it ends when he cannot do more for the patient right now. The task contains many kinds of subtasks. The clinician decides what to do and in which sequence.

Work area 2: [Patient treatment]

This work area comprises …

C10. [Perform clinical session]

[A clinical session may comprise diagnosis, planning of treatment, actual treatment, evaluation, etc. Usually several of these are carried out, but it may also happen that only planning, for instance, is carried out.]

Start: [Contact with the patient or a conference about the patient.]
End: [When nothing else is to be done about the patient right now.]
Frequency: [In total: Around 15,000 per day. Per user: A maximum of 20 per day.]
Difficult: [Disasters with many injured. (Better describe it as a separate task. See the guide.)]
Users: …

Subtasks and variants:	Example solutions:	Code:
1. [Identify the patient.]	[The system can read an electronic bracelet, e.g. for unconscious patients.]	
2. [Assess the state of the patient. See open diagnoses and the related indications. See notes. See results of services ordered earlier and compare them with expectations. (The data to overview comprises D1-D4).]	[The system shows an overview of everything on one screen, e.g. with a Gantt-like time dimension. It is possible to drill down to details from the overview].	
3. [Provide services that can be given on the spot, e.g. local services such as blood pressure and SAT.]	[The system makes it easy to record the results on the spot.]	
4. [Follow up on planned services and results. Check for violated deadlines.]	[The overview shows ordered services and their state, e.g. deadline violation.]	
5. [Adjust diagnoses (modify, add, delete, prioritize). Check against standard recommendations. Write notes.]	[The system makes it easy to change diagnoses and write notes on the spot.]	
5p. [Problem: Cumbersome to see standard recommendations.]	[The system can show recommendations and checklists based on selected diagnoses.]	
6. [Plan and order new services. Check against available time for all parties - including the patient. (See the long subtasks C11 to C… for prescription, booking, …).]	[The system makes it easy to request services on the spot. For bookings, the system shows available dates and times for all parties.]	
6p. [Problem: Parts of the request are forgotten.]	[The system can use standard packages of services.]	
6q. [Problem: Errors when data is written on paper and recorded later.]	[The system makes it easy to record on the spot.]	
7. [Maybe discharge the patient. (See task C6).]		

C11. Prescribe medicine - a long subtask

Sometimes there are so many subtasks in a task that the description becomes hard to overview.

One solution is to bundle the subtasks into logical groups with headings. We have seen this work fine with 50 subtasks. The purpose of the bundling is only to help the reader. The subtasks may still be carried out in almost any sequence.

Another solution is to make each bundle a **long subtask** with a separate C-number. As an example, subtask C10-6, *plan and order new services*, refers to several long subtasks: C11, *prescribe medicine*, C…, *booking*, etc.

C11 is shown in detail. Notice that a long subtask doesn't have its own start and end description. It is simply a part of the main task. However, it makes sense to specify the frequency because only some clinical sessions have prescriptions or bookings.

Subtask 6, *Calculate dose,* shows how business rules can be embedded in a task. We might split the rules into several subtasks, but as stated we leave this to the supplier.

C18. Perform clinical session - in another environment

It may happen that a task is carried out in different environments with different needs for IT support. One example is the clinical session (C10) when the medical staff is moving around from patient to patient. The customer would like to support it through PDA's or Smartphones. In theory all we need is to state in the task introduction that it may also be a mobile environment.

However, where should the supplier specify his solution, which is probably different from the normal PC support? And how will the customer assess the solution? The suggestion is to define a task for each environment:

C10: Perform clinical session, stationary.
C18: Perform clinical session, mobile.

What about the long subtask C11, *prescribe medicine*? To make sure that it too is supported well in both environments, we should repeat it.

As for C2, *admit immediately*, don't worry about the same subtasks appearing in several tasks. We need to check the support of them in the different contexts.

Avoid use-case-like tasks

If you use the task concept correctly, there will be rather few tasks to describe. Many large systems can be described with just 10-30 tasks. This is an advantage because you get a better overview and have much less to write.

We often see requirement specifications where 10 tasks have been expanded to around 100 **use cases**, each of which takes up one or more pages, although little happens in each of them. The cause is usually that each subtask has been specified as if it was a separate task with start and end, frequency, etc. In real life the use cases are not separate but done in combination with other use cases until "coffee break". When described in the use-case way, the supplier gets no feel for how the use cases relate to each other, and as a consequence he cannot support them well. Here is an example from the hospital world:

C11. [Prescribe medicine for the patient (long subtask)]

[This is not a separate task but a long subtask carried out during a clinical session. (For this reason "start", "end", and "user" are unnecessary.]

Frequency: [In total: Around 30,000 times per day. Per user: A maximum of 20 times per day.]

Subtasks and variants:	Example solutions:	Code:
1. [Assess the entire medication pattern of the patient, in this admission as well as other admissions.]	[The system shows an overview of all medications, CAVE, diagnoses, etc.]	
1p. [Problem: Cumbersome to see standard recommendations]	[The system can show recommendations and checklists based on diagnoses and drug type.]	
...		
6. [Calculate dose. Check that it is reasonable. Check for interaction with other drugs.]	[The system offers a calculation based on the recorded body weight. It checks for interactions.]	
6p. [Problem: Translation between various units. There may be a difference between the unit of prescription (e.g. mg) and the unit of dose (e.g. number of tablets).]	[The system shows the dose in prescription units as well as dose units.]	
...		

...

C18. [Perform clinical session, mobile]

[Clinical sessions may be performed when medical staff is moving around from patient to patient, e.g. with a PDA, tablet or mobile phone. In principle we have the same subtasks as in C10, but they cannot be supported in the same way. In order to allow the supplier to specify his solution for the mobile situation, we repeat the clinical session task here.]

Start: When …
End: When …
Frequency:

A harmful specification from real life - 3 pages in total

Use case 2.1. Show diagnoses

The clinical user wants to obtain an overview of the patient's diagnoses and their relationships.

Start: The user wants to inform himself of the development in the patient's state of health.
End: …
Precondition: The user is logged in. The patient is recorded and selected.

Step:	Example solution:
1. Show the hierarchy of diagnoses.	
2. Select display mode.	E.g. a hierarchy or a Gantt diagram.
3. Select the level of detail.	E.g. expand or collapse with plus and minus.
4. Show notes about a selected diagnosis.	
5. Show date and author for the note.	
6. Show possible external causes of the diagnosis.	
. . .	

This is not a true task because it isn't closed in the coffee-break sense. It will be part of a larger task, for instance a clinical session. Furthermore it has so many details that it almost prescribes a specific user dialog. Notice the computer-focused trigger: *The user wants to . . .* It is an indication that it may not be a true task.

The reader cannot see the purpose of this use case. Why does the user want this overview? To find a treatment of the patient, to explain a new symptom, or to write a report about the patient? Although the user may get support for this use case, it is not sure he gets support for the larger task. He might for instance have to write down the diagnoses on paper in order to carry out the next step of the true task.

Don't describe data as subtasks

The use case above is 3 pages in total. One reason is that the analyst has tried to describe data as steps. Notes, dates and external causes are handled as separate steps. The real specification also had use cases *Create diagnosis* (4 pages) and *Change diagnosis* (3 pages). They referred to almost the same data. It was hard to ensure consistency, of course.

The solution is to describe data separately, as we do in Chapter D. From the sub-tasks you may briefly refer to the data that are relevant in this subtask. The template shows examples in C1-3 and C10-2.

Sometimes it is useful to list the necessary data more precisely, for instance in a single subtask or as a requirement note below the task.

Tasks have no preconditions

The use case above has two preconditions: The user must be logged in, and the patient recorded and selected. This enforces a flow between use cases. The user must first carry out the login use case, next the select patient use case, then the show diagnoses use case.

Tasks don't have preconditions, but the subtasks may have, although we rarely need to write them. The clinician can start a clinical session at any time without any precondition. It is part of the task to identify and select the patient (subtask 1). It is an implicit precondition for the remaining subtasks that this has been done. Since the context is clearly visible, there is little reason to write an explicit precondition for all of these subtasks.

What about the login precondition? In a task perspective this is not a demand but a solution to a problem: who is the user and what is he allowed to do? Login is only a cumbersome way to do this. The template deals with these issues in section H, security and doesn't mention them in the tasks.

Flows and BPMN - overview, not requirements

It is often useful to show the total course of events - the task flow, also called a business process or a high-level task. It might be the course of events for treatment of a patient. Or it might be the life cycle of a treatment type from the time it was investigated by a health commission until it many years later is canceled by another commission.

It may be a good idea to describe the task flows in the requirement specification, for instance at the beginning of Chapter C, but it must be clear to the reader that they are purely informative and not requirements.

Flows can be described graphically or as text. A widely used graphical notation is BPMN (Business Process Modeling Notation) that shows each step as a node and connects the nodes with arrows that show what comes next. This can provide a great overview - unless you go into too much detail trying to specify also the flow when something goes wrong.

Here is a flow described as text: the business process for a patient.

Flow 1: **Patient treatment**
Start: The patient is referred to the hospital from a practitioner or arrives in emergency.
End: The patient is cured.

Step:	Solution:
1. Admit the patient.	See C1 and C2.
2. Make a diagnosis.	See C10.
3. Plan the treatment.	See C10.
4. Carry out the treatment.	See C10.
5. Assess the result.	See C10.
6. Discharge the patient.	See C… and C…
7. Follow-up treatment at home.	**?**

This is the logical, ideal course of events. When you describe a flow, you often detect new demands for IT support. In this case we detected the need for coordination with home treatment (shown as a question mark in step 7).

Notice that the logical flow steps have a many-to-many relationship to the physical tasks. Step 1 is handled by two tasks C1 and C2 (admit patient before arrival and admit immediately). In contrast, steps 2 to 5 are handled by only one task, the clinical session.

Are the true tasks really like this? Yes, in particular in slightly complicated cases. Elderly patients, for instance, often have several diagnoses and staff try to treat all of them during the same admission. This means that a single clinical session can make a new diagnosis, treat an earlier one, and assess the result of another treatment. Task C10, *perform clinical session*, expresses this in a compact way which emphasizes overview of the entire situation and specifies that many activities may be carried out at the same time.

D. Data to record

This chapter describes the data to be stored in the system. Data may be described in several ways. The template shows five ways: (1) a short textual description of the classes (tables), (2) an E/R diagram (Entity/Relationship model), (3) a data dictionary with details of each field, (4) the contents of some existing tables, (5) the contents of existing screens.

The E/R diagram shows data classes as boxes. The name of a class should be singular, i.e. *Person* rather than *Persons*. Think of the box as a pile of file cards, each holding data about a single person. What you see is the card for one single person.

A crow's foot shows that one object (entity) relates to several objects in another class. As an example, one person's file card is related to several admission cards (strictly speaking to zero or more cards). Reading the crow's foot the other way, one admission card is connected to only one person card. A crow's foot may be a smooth curve.

When the data are in a relational database, a class corresponds to a table. However, E/R diagrams are also very useful when data are not in a database.

The diagram lists the fields (attributes) outside the box to save space and improve overview. In many cases we show only some of the fields.

A UML class model is very similar, but fields are shown inside the boxes and connectors are lines with cardinality shown as 0:1, 1:*, etc. When a line cannot be straight, it is broken at right angles. These seemingly small differences make a huge difference when you try to get an overview of a large diagram. Your vision system can much easier overview an E/R diagram than a UML diagram. Further, a UML diagram often needs five times as much space.

D. Data to record

The system must record the data described in this chapter. The user can create, view, and change the data through the tasks described Chapter C. In many cases data has to be exchanged with external systems as specified in Chapter F.

Figure 3 is an Entity/Relationship diagram (E/R) that gives an overview of the data. Each box holds all entities (records) of a certain kind. [As an example, D1 holds all patient diagnoses.] A crow's foot shows that an entity in one box is related to many entities in the other box. [As an example, each diagnosis is related to one diagnosis type while each diagnosis type is related to many diagnoses.] Data need not be structured in this way in the system, but it must be handled in some way.

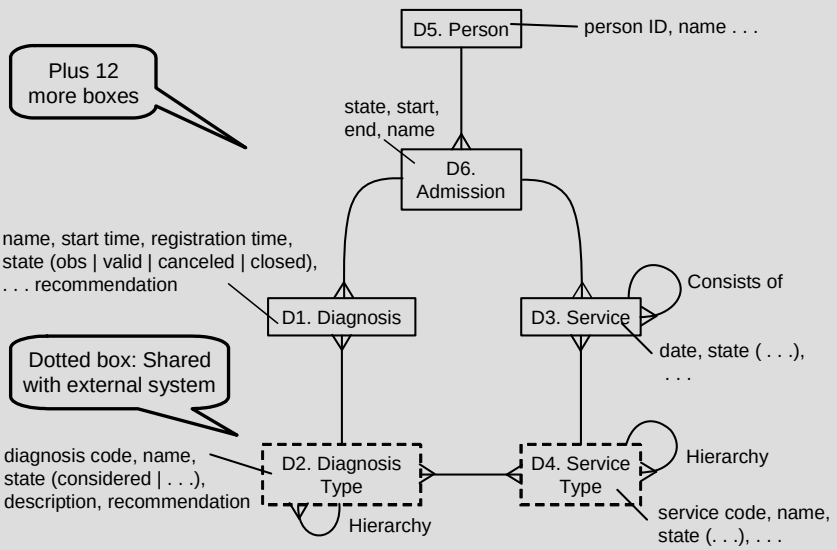
The dotted boxes show data that are (partly) shared with an external system through system integration (Chapter F). Here is a brief explanation of the boxes of the diagram.

D1. [Diagnosis: Holds a record for each patient disease, actual as well as suspected. They correspond to the National Health Classification (NHC), but there is also a need for recording diseases that are not in NHC or cannot be classified until later.]

D2. [Diagnosis type: A catalogue of all possible diagnoses - independent of the patient. The users will select the patient diagnoses from this catalog. A diagnosis type specifies the name and NHC code (where possible), recommendation, standard treatment packages (through the relationship to the catalogue of service types), and more. The catalogue is updated by the customer based on medical experience and through data transfer from the National Health Organization.]
…
D5. [Person: Holds a record for each patient, staff and other related persons with name, address, etc.]

D6. [Admission: Holds a record for each admission with admission date, department, etc.]
…
D18. [Municipality: A catalogue of all the municipalities.]

Figure 3. Data model for the system



D1. Diagnosis

This section is the data dictionary for the diagnosis class. It consists of these parts:

1. The number and name of the class. Classes are numbered D1, D2, etc. To avoid too much renumbering during analysis, you may bundle the classes and start each bundle with a round number.

2. Examples of what an object might be. Show typical as well as unusual examples.

3. The source of the data. Where does it come from? It might be entered during a task, collected by the system, or imported from another system. In many cases you can describe it for all fields at the same time; in other cases some fields need a description of their own.

4. The use of the data. It may be used in tasks or exported to other systems. Again there may be a common description for all the fields or separate descriptions for some fields.

5. The data volume. This is a requirement in a table. The system must be able to store this amount of data. The requirement also specifies that a history trail is needed. Section L3 specifies for how long time the data must be kept and how fast archived data must be retrieved.

In the example, the data volume is given as the number of new diagnoses per year. This also gives us the number of create-transactions per day, and an indication of the number of create-transactions in peak load periods. This is important for stating response time requirements in L1.

6. A table with details for each field and each connector (relation) to another class. Attributes are numbered sequentially. Problems associated with an attribute are numbered p, q, etc. The list has three columns, similar to tasks.

The template example is written on a high level. In many cases details such as date format and text lengths may be needed too, for instance in the solution column as shown for D1-4. In some cases a specific format is necessary, and then it is a requirement in column 1. Use it sparingly; it reduces the chance of finding a COTS system that matches the requirement.

Notice that problems, requirement notes and solution notes may be used as for tasks.

D1. [Diagnosis]

[A diagnosis is a disease or a symptom for a specific patient.]

Examples: [There is a fuzzy distinction between diseases and symptoms. As an example, cholera as well as coughing are "diagnoses".]
Data source: [Diagnoses are recorded during clinical sessions (C10) and often during admission (C1).]
Data use: [Diagnoses are shown in patient overviews, for billing and for government reporting.]

Data volume:	Example solutions:	Code:
1. Around [800,000 diagnoses] are recorded a year. All changes must be kept in a history trail.		

Fields and relationships:	Example solutions:	Code:
2. [Diagnosis Code: Relation to Diagnosis Type. The patient's primary diagnosis may change during the admission. The primary diagnosis is used for billing and government reporting.]		
2p. [Problem: Very hard to select the right NHC code from the 20,000 possible ones.]	[See solution notes below.]	
3. [AdmissionID: Relation to the Admission, which in turn refers to the patient (Person).]	[The system records it automatically based on the currently selected patient.]	
4. [Name: Usually the name from Diagnosis Type, but may be a name entered for this specific patient.]	[Field length: 100 characters.]	
5. [State: A diagnosis may be in these states: Obs, valid, canceled, closed.]		
6. [Start Time: The date and time from which the diagnosis is in this state. Usually it is the same as the Recording Time, but not always, e.g. if you record that the patient started coughing yesterday.]	[The system makes it easy to choose the Recording Time as the Start Time.]	
7. [Recording Time: All changes to the diagnosis are recorded and the earlier versions are kept in the system. Usually, doctors are only interested in the latest version.]	[The system records it automatically.]	
...		
17. [Recommendation: The recommendation valid at the time of creating the diagnosis.]		

Solution notes

[The user might for instance select a diagnosis code in these ways:
a. Browsing a conceptual hierarchy (corresponding to the NHC super and subclasses)
b. A reduced hierarchy so that the department as a default see only the diagnoses relevant for them.
c. "Live search" where the user enters part of the diagnosis name, and the system shows possible matches keystroke by keystroke.]

D2. Diagnosis type

The diagnosis table D1 holds the actual diagnoses for the patients. In contrast, D2 is an example of a type table. It holds the catalogue of all possible diagnoses.

It is usually important to specify also the type tables, particularly when the system must be able to add a type, change it, and maybe keep track of the history of each type.

Notice how D2-6 deals with the length of the description field. It should be around two lines, but the exact number is not important. For this reason the customer has written a suggested length in the solution column. The supplier may adjust it to what is convenient for him, for instance 255 characters.

D3. Service

There are many subclasses of service in an EHR system. It is hard for the customer to specify all of them. In the example, the customer has specified the common fields and relations that all services have.

Section D3.1 specifies the services that are clinical measurements. The special fields are defined through a screen cut listing the fields in his current system. (The customer should add the existing field lengths and formats.)

Section D3.2 specifies the services that are surgery.

Section D3.3 specifies the services that are patient medication. In this case, the customer didn't have the table formats, but used screen cuts from his existing medication system. This requirement gives the supplier some basis for judging what the customer needs, but there is a risk of misunderstanding and later conflicts.

D2. [Diagnosis Type]

[The collection of diagnosis types makes up the diagnosis catalogue.]

Examples: [DA009: Cholera without specification; DR059: Coughing.]
Data source: [Imported from the NHC web site.]
Data use: [The user selects a diagnosis type when recording a patient diagnosis.]

Data volume:	Example solutions:	Code:
1. [There will be around 30,000 diagnosis types. NHC has presently around 20,000 types.]		

Fields and relationships:	Example solutions:	Code:
2. [Diagnosis code: NHC code (similar to ICD10) or a temporary code.]		
3. [Name: The full name of the diagnosis, e.g. "Cholera without specification".]		
4. [State: A diagnosis type can be in one of these states: Considered, valid, outdated.]		
5. [Parent: Relation to a more general diagnosis type in a conceptual hierarchy. Example: A022A Salmonella Arthritis belongs to A02 Salmonella.]		
6. [Description: A longer text, but not more than one or two lines. Even longer descriptions may be found in the "Recommendation".]	[Field length: 160 characters.]	
7. [Service types: Relation to service types that may be used to treat this diagnosis.]	[The system may extract the information from the Recommendations.]	
...		
10. [Recommendation: A long text describing indications, medical practice, etc.]	[Might be a URL.]	

D3. [Service]

[A service is something measured or given to the patient. There are many subclasses of service, e.g. measurements, surgery and medication. At present they are stored in separate tables or even in separate systems.]

Fields and relationships common for all services:	Sample solutions:	Code:
1. [Service code: Relation to Service Type.]		
2. [AdmissionID: Relation to the Admission, which in turn refers to the patient (person).]	[The system records it automatically based on the currently selected patient.]	
3. [Date: The date the service was given.]		
4. [State: In the normal flow a service may be in these states: Ordered, confirmed (by the service provider), started (e.g. sample taken), completed, assessed (by the clinician). Exceptionally, the state may be: Canceled, changed.]		
5. [Consists of: Relation to services that are part of this service, e.g. surgery that consists of several treatments.]		

D3.1. Patient measurement

Examples: [Blood pressure; Body Weight; B-glucose; Gamma globulin; X-ray.]
Data source: [Some are recorded during a clinical session; others are imported from an external system, e.g. lab results.]
Data use: [Used in patient overview and detail view to support diagnosing and treatment.]

Data volume:	Example solutions:	Code:
1. [Around 100,000 measurements are recorded a day. Of these 5,000 are pictures. All changes must be kept in a history trail.]		

Fields:	Example solutions:	Code:
2. [A patient measurement should include the data from the present table. See Figure 4, tblPatient-Measurement. Notice that the present table doesn't have the common fields: admissionID and state.]		

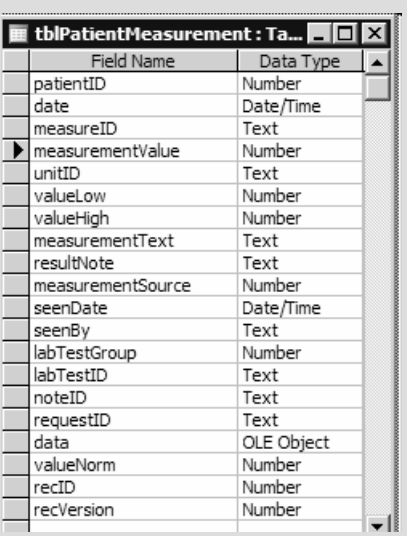
D3.2. Patient surgery

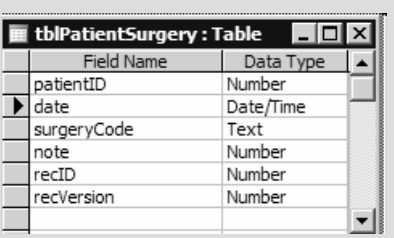
Examples: [Heart Bypass Operation; Photodynamic Therapy (PDT).]
Data source: [Recorded during and after surgery.]
Data use: [Used in patient overview and detail view to support diagnosing and treatment.]

Data volume:	Example solutions:	Code:
1. [Around 100 surgeries are recorded a day. All changes must be kept in a history trail.]		

Fields:	Example solutions:	Code:
2. [A patient surgery record should include the data from the present table. See Figure 4, tblPatientSurgery. Notice that the present table doesn't have the common fields: admissionID and state.]		

[Figure 4. Present service tables]





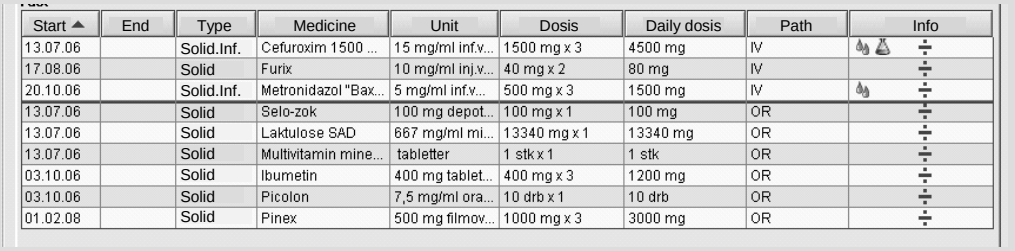
D3.3. Patient medication

Examples: [Ibumetin, 400 mg*3; Furix, 40 mg*2.]
Data source: [Recorded as prescriptions during clinical sessions.]
Data use: [Used in patient overview and detail view to support diagnosing and treatment.]

Data volume:	Example solutions:	Code:
1. [Around 30,000 prescriptions are recorded a day. All changes must be kept in a history trail.]		

Fields:	Example solutions:	Code:
2. [A patient medication record should include the data that the present system shows. See Figure 5, screen shot from the present medication system.]		

[Figure 5. Present medication data]



Start	End	Type	Medicine	Unit	Dosis	Daily dosis	Path	Info
13.07.06		Solid.Inf.	Cefuroxim 1500 ...	15 mg/ml inf.v...	1500 mg x 3	4500 mg	IV	
17.08.06		Solid	Furix	10 mg/ml inj.v...	40 mg x 2	80 mg	IV	
20.10.06		Solid.Inf.	Metronidazol "Bax...	5 mg/ml inf.v...	500 mg x 3	1500 mg	IV	
13.07.06		Solid	Selo-zok	100 mg depot...	100 mg x 1	100 mg	OR	
13.07.06		Solid	Laktulose SAD	667 mg/ml mi...	13340 mg x 1	13340 mg	OR	
13.07.06		Solid	Multivitamin mine...	tabletter	1 stk x 1	1 stk	OR	
03.10.06		Solid	Ibumetin	400 mg tablet...	400 mg x 3	1200 mg	OR	
03.10.06		Solid	Picolon	7,5 mg/ml ora...	10 drb x 1	10 drb	OR	
01.02.08		Solid	Pinex	500 mg filmov...	1000 mg x 3	3000 mg	OR	

E. Other functional requirements

Most of the system functionality is simple data creations, deletions, edits and queries that are implicitly required to support the tasks and system integrations. This chapter describes functionality that is more complex.

E1. System generated events

The system may do things on its own, for instance collect data from the environment or send reminders to users when time limits are exceeded.

Requirement E1-1 asks for a reminder when an admission has been "forgotten". There must be a task that handles this reminder. In the example, task C1 *Admit patient* deals with it as one of the possible triggers.

Requirement E1-2 asks for a reminder when a Labsys service has been lost. Here too there must be a task that handles this reminder. This task is not mentioned in the template. It is carried out by a department secretary or the chief nurse.

E2. Reports

Often the existing system can print heaps of reports, but for most of them the customer doesn't know whether they are used and for what. The template shows how to transform this lack of knowledge into requirements.

Report 1 has a well-defined purpose and we can describe the format precisely, for instance through a sample print.

Report 2 has a well-defined purpose, but no specific format. It is useful to refer to the task or tasks where this report is used to help the supplier understand what is convenient.

Report requirement 3 gets round the lack of knowledge by asking the supplier to offer a fixed price per report. In this way the customer can delay the decision on which reports are needed. The fixed price prevents the supplier from abusing the de-facto monopoly he has got after signing the contract. The price may depend on the complexity of the report, for instance expressed as a price per Function Point or the modern version, COSMIC point.

Requirement 4 gets round the problem in another way by asking for a report generator. It will allow the customer to develop his own reports. The example asks the supplier to specify how easy it is to develop the reports, for instance by stating what kind of users can do it and how much training they need.

Requirement 5 states that all reports must be available on the screen as well as in print.

E. Other functional requirements

Most system functions are simple creations, deletions, edits, and queries that need no further specification. They are implicitly given by the task descriptions (Chapter C) and the data descriptions (Chapter D). In addition, the system must be able to perform the functions specified in this chapter.

E1. System generated events

The system must generate these reminders:	Example solutions:	Code:
1. If an admission has been parked for x days the central admission office must be reminded. System administration must be able to define x.	X is typically 4 days, but may vary between departments.	
2. If a Labsys service has been ordered but not completed within 24 hours, the clinicians must be reminded.		

E2. Reports

Some reports are needed in connection with the tasks described in Chapter C. The report formats are not essential as long as the tasks are supported well. These reports are not described here. There is also a need for reports with ad hoc purposes, cross-task purposes, and reports with a precise format. They are specified here.

Report requirements:	Example solutions:	Code:
1. [Checks must be printed on preprinted forms with the format shown in …]		
2. [The system must be able to show an overview and forecast of the bed occupation (used for instance in task …).]	Figure … shows an example of such a report.	
3. The supplier must develop up to 100 new reports at a fixed price as part of the maintenance.	The price per report is ____. (The price may depend on the complexity.)	
4. The system must contain a report generator that is easy to use.	How many of the staff will be able to develop the reports in appendix X after a course of __ days: type 1 / type 2 ordinary users __% / __% super users __% / __% the customer's IT staff __% / __%	
5. The system must be able to show all reports on the screen as well as on print.		
…		

E3. Business rules and complex calculations

Rules and computations may be described in several ways. Some fit nicely into task descriptions, for instance this subtask in C11, Prescribe medicine:

> *Check that the medicine doesn't interact with other drugs the patient takes.*

Other rules are part of the data requirements (e.g. possible states of a service) or security rules (e.g. who has the right to do what?). This section specifies more complex rules.

Function 1 in the example requires a computation that is described in a separate appendix (*waiting list calculation*). The appendix may for instance contain an algorithm described as a small program, a flow chart, or a table of the possibilities.

Function 2 refers to a public document where the rules are described (*salary agreements*). In order to translate this into a solution, the supplier needs a lot of expertise in the salary domain.

You may also indirectly specify a function through an accuracy requirement, for instance that the system must be able to recognize human speech with a back-ground noise of 30 dB. Or that the system must be able to calculate a duty roster that is at most 3% more expensive than the optimal plan.

Function 3 shows a rule expressed as a state-transition diagram. A diagnosis for a specific patient can be in one of these states: *obs, valid, canceled, closed*. Officially, it can only change state as shown by the arrows. User actions cause all these state transitions, except deletion of the diagnosis. Deletion is done automatically after 20 years. As requirement E3-3 explains, users should be able to make any state change anyway (except for undoing the final deletion).

Function 4 shows a more complex rule as a state-transition diagram. It specifies how the state of a Labsys request changes as a result of user actions as well as messages sent by Labsys. During a state transition some actions may take place. The diagram shows them as dotted lines branching off from the transition arrow. In this example, the actions are to send messages to Labsys.

Diagrams such as these can be detailed further with activity diagrams (from UML) or SDL (from the telecommunication industry). Sometimes this level of detail is important, but in most cases it specifies a solution rather than a user demand. In the example, the user doesn't really care about these Labsys details, but it is important to him that he can see how far the Labsys request has come. This could be stated as the requirement.

E3. Business rules and complex calculations

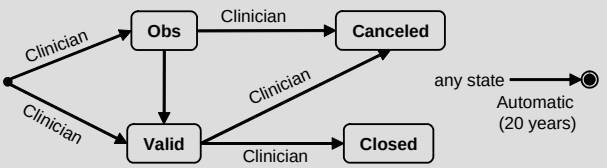
Some business rules are explicitly specified in the task steps, e.g. *Check that* … [(example in C11-6)]. Other business rules are explicitly specified in the data descriptions [(example in D3-4)], and some are specified as access rights (section H1). Here are additional business rules and complex functions.

Function:	Example solutions:	Code:
1. [Waiting list priority must be calculated as described in …]		
2. [Salary calculations must at any time follow the collective agreements (see also the maintenance requirements in …).]		
3. [Normally, a diagnosis may only change state as described in Figure 6. In case of mistakes, the user must be able to deviate from the rules (see also H4-2).]	A user who tries to deviate from the rules will be asked whether it is intentional. If so, the change is made and logged in …	
4. [Inside the system, a service requested from Labsys changes state as described in Figure 7.]		

Requirement note: State-transition diagrams

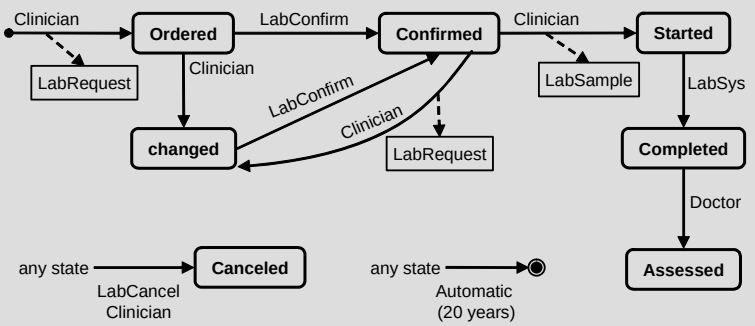
[Figure 6 shows that a clinician creates the diagnosis. It is created in either state *Obs* or state *Valid*. Clinicians can change the state further according to the diagram. The diagnosis disappears when the system automatically cleans up the data after 20 years.]

[Figure 6. Diagnosis states]



[Figure 7 shows how the state of a Labsys service changes inside the system. A clinician creates a Labsys service in state *Ordered*. During the creation, the system sends a LabRequest to Labsys. When Labsys sends a *LabConfirm* message to the system, it changes the service state to *Confirmed*. A clinician takes a sample from the patient, sends it to the lab and tells the system, which changes the service state to *Started*. The service can change state in other ways as specified in the diagram.]

[Figure 7. Labsys service states and messages]



E4. Expansion of the system

In some cases the customer needs to be able to expand the system himself in some areas. He may for instance want to experiment with new screens to improve usability, or he may fear that the supplier will charge an unreasonable price for expansions.

This section asks for functionality that will make some kinds of expansion possible without involving the supplier. Some years ago, suppliers were reluctant to allow such things, because they feared for the correctness and stability of the system. This has changed and even ERP systems such as SAP and Axapta provide better and better possibilities for expanding the system.

In the EHR example there is a significant demand because there are more than 20,000 types of patient service, each with their own data fields; and the number grows steadily. It is not acceptable that the supplier is needed for changing the system whenever a new type of service is introduced. Similarly, many medical specialties have their own needs for data visualization.

There is also a demand for future integration with external systems. This is handled in section F0.

Notice that the template not only asks for expansion functionality, but also for the rights to use it. This is based on bad experiences with suppliers who provide the functionality but keep the rights for using it and for extracting the data stored in the system.

E4. Expansion of the system

[The system shows and maintains data through the user screens. The customer expects that he or a third party is able to modify the screens and add new ones in order to create overview for medical specialties, new work procedures, etc.

The system handles many types of medical services, often with special combinations of data. The customer expects that he or a third party can add new types of services. In this section, "customer" means the customer's own IT staff or a third party authorized by him.]

Expansion requirements:	Example solutions:	Code:
1. [The customer can define new types of services and corresponding screens within the limits defined by the data descriptions in Chapter D.]		
2. The customer can define screens that combine data from the entire data model in Chapter D (arbitrary views of data).		
3. [A screen can activate functionality in the EHR system and in external systems integrated with the EHR system.]	[E.g. request of a service, notification of medical staff, print of a report.]	
4. A screen can be composed of many types of components (controls) and their color can reflect data values.	E.g. text boxes, tables, buttons, graphs, pictures.	
5. The customer can add new types of components for use in the screens.		
6. Screens can be defined for several kinds of equipment.	E.g. PC, PDA, tablet, Smartphone.	

Documentation and rights:	Example solutions:	Code:
7. The tools for composing screens, adding new component types, etc. must be documented in such a way that the customer can understand them and use them for the intended purpose.	A course of ___ days is necessary to use the tools.	
8. The customer must have the right to use the tools and extract the data stored in the system.		

F. Integration with external systems

The trend is that new systems must be integrated with more and more other systems - **external systems**. More than ten external systems are quite common.

In some cases we can avoid explicit integration requirements because full support of the tasks requires integration. We did so in C1-3a (use of MedCom for data transfer). Usually, however, integration is a complex affair, and it will be hard to evaluate a supplier's integration solution by trying to carry out the tasks. It is particularly difficult if we want to make an early proof of concept (B2). So usually we need explicit integration requirements.

It is hard to specify integration requirements. The external systems exist and the supplier has to know about their technical interfaces (API's or XML services) in order to estimate his integration costs. Yet the customer rarely has this information.

SOA or data replication?

Some customers listen to the IT gurus and ask for a Service Oriented Architecture (SOA) where systems connect with XML services and data are only stored in their source system. Other systems retrieve it from there. In principle it is a great idea, but the customer doesn't realize that this requires 10-50 times more computer time than traditional approaches. It also makes it impossible for the supplier to ensure fast response times and high operational availability, because his system depends on other system's response times and availability.

When the supplier offers a COTS-based system, it may become a really expensive solution for other reasons too. The COTS system retrieves data from its own database, but now it must be retrieved through SOA. The supplier must change his system in hundreds of places - even if it is nicely made with a multi-layer architecture. A system that has been changed in so many places cannot be maintained as part of maintaining the COTS system. So maintenance will also be very costly.

An alternative solution is to replicate data across systems and synchronize data periodically or at demand. This is usually much easier to add to a COTS system.

Which system should initiate the synchronization? It depends on what is possible with the existing systems. And the customer shouldn't care. He should only ensure that his demands are met. So what are the real demands? A study of many system integrations shows that several aspects are involved:

a. Data recency: How old is the data that the system shows? This is the key concern in integration. With a SOA architecture, the data on the screen will be a few seconds old. With a replicated solution it may be hours or weeks old, but often this is sufficient (the template shows examples).
b. Task support: Can the user tasks be supported well with this integration?
c. Data to transfer: Which data - to or from the external system - or both ways?
d. Data protection: Avoid data loss, duplication, and security breaches.
e. Other functionality: Can the system order other functions in the external system, for instance remind users or print data?
f. Documentation and rights: What to document? Who may use it for what?
g. Responsibility: Who will make and test the integration and how will the "other end" help? In spite of what the gurus say, there is usually a need to make changes at both ends.

F. Integration with external systems

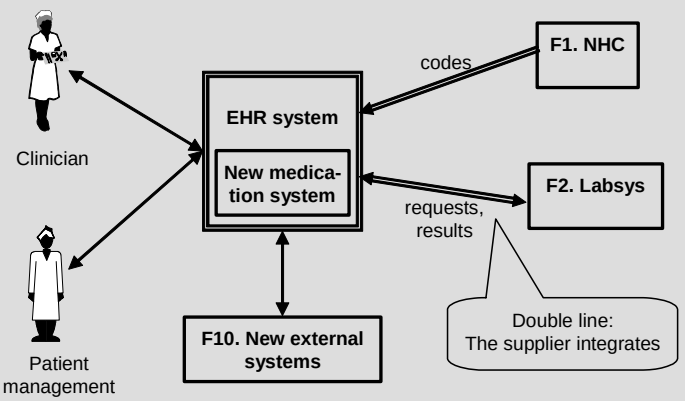
The system must integrate more or less closely with the external systems shown in Figure 5 (context diagram). Double-line arrows show integrations that the supplier is expected to deliver. The integration comprises data sharing and the ability for the user to activate functionality in other systems (external systems).

In this Chapter, "customer" means the customer's own IT staff or a third party authorized by him.

System data (S-data) are the integrated data stored locally in the system (S).
External data (E-data) are the integrated data stored in the external system (E).

Here is a short explanation of the external systems:
F1. [NHC: The National Health Classification system. The National Health Organization updates it regularly.]
F2. [Labsys: The customer's present lab system for …]
F3. …
F10. A new external system that the customer will buy later.

[Figure 8. Context diagram]



The template has sections and examples for each of these aspects.

The template starts with a verbal overview of the external systems and a graphical overview in form of a context diagram. It is similar to the context diagram in the background section of the template, but it will usually contain more details, for instance the system codes F1, F2 . . .

Show the system to be delivered as a box with double-line borders. Show the integrations to be performed by the supplier as double-line arrows. Let the arrows point in the direction data move. Label each arrow to indicate the data that flow.

In the example, the supplier has to integrate with the existing NHC system and Labsys. Note that he is not required to integrate with new external systems. Someone else may do it.

F0. Common integration requirements

This section covers requirements that apply for all the integrations unless something else is stated.

F0-1 requires that data may only be transferred to the user's PC if he is allowed to see them. So the system may not depend on only special PC programs showing the data. It would be too easy to install a program that peeks.

F0-2 to 4 require the system to protect against technical problems with lost or duplicated data.

F0-5 specifies that the customer (or a third party) must be able to migrate the data to another system. This is a key requirement for being able to switch supplier later. Surprisingly many customers forget this and the supplier gets a monopoly.

F0-6 to 9 specify that the customer must be able to integrate the system with other systems. He must have the means, documentation, and rights to do so, and the supplier is obliged to support the work. If all the external systems had met similar requirements, integration would be much simpler.

Notice how it is possible to verify the quality of the documentation by asking a typical third party software house to try out the documentation. This should be done early in order to make it likely that this kind of documentation will suffice for third party expansion of the system (see section B2).

F0. Common integration requirements

The requirements in this chapter apply for all the integrations unless explicitly stated.

Access rights to data:	Example solutions:	Code:
1. The system may only transfer E-data to the user's PC when the user has the right to see it according to H1.		

Protection of data:	Example solutions:	Code:
2. The system must protect against loss or duplication of data transferred between the systems, e.g. because one or both systems have been off-line or closed down.		
3. The system must protect against concurrency problems, e.g. that user A sees and then updates E-data, while user B does the same. Neither A nor B will notice the conflict.		
4. To help error tracing, the system must log all transfer errors.		

Migration rights to data:	Example solutions:	Code:
5. The customer must have the means and rights to extract and use all data described in Chapter D, e.g. for converting the data to another system.		

Integration rights and documentation:	Example solutions:	Code:
6. The customer must have the means and rights to integrate the system with other systems.		
7. It must be easy to add new interfaces, e.g. SOA services, database queries, or API's.	The customer can add the interface. Or: The supplier can do it at a fixed price.	
8. The technical interfaces to S must be documented. The documentation must be understandable to a typical software house and found suited for integration and data retrieval.	A course of ___ days is necessary to use the documentation and make the integration. Documentation samples must be delivered early, preferably as part of the proposal (cf. B2-4).	
9. The supplier must loyally support the customer in the integration or migration effort with qualified staff at a fair price.		

F1. NHC

This section is an example of a very loose integration with an existing system, NHC, the National Health Organization's classification codes. NHC has code files that anyone may download.

The introduction outside the tables gives the assumptions for the requirements, similar to the assumptions for tasks descriptions.

Tasks: Which tasks utilize the integration?
E-documentation: How to get the documentation of the external interface?
E-data updates: How frequently are NHC codes updated inside the NHC system?
Data volume: How much data is transferred?

F1-1 specifies that the supplier has to make the integration. It is assumed that he doesn't need support from someone else to do it (a reasonable assumption in this case).

There are no special requirements for task support. The introduction says that the data are used in most tasks. It is sufficient in this case.

F1-2 specifies the data to be transferred from NHC.

F1-3 shows that the recency of data is not urgent. If the system has the data one week after they have been released by NHC, everything is okay. The example solution mentions that a periodic transfer is sufficient. The transfer might also be started manually by IT support when the health authorities announce the changes.

F1-3p mentions an existing problem about conflicts between local codes and new official codes, and suggests two solutions.

F1-4 mentions that more recent data are needed sometimes.

There are no requirements for a specific response time (how fast the transfer is). The system is not required to use other functions in NHC or transfer data to NHC.

F1. [NHC]

External system: [The NHC tables comprise codes and corresponding names for diagnoses, services, health departments, etc.]
Tasks: [The codes are used in most of the tasks. However, the department codes are retrieved from another system.]
E-documentation: [The tables are publicly available from the web site of the National Health Organization. They are zip text files with fixed field spacing. They are documented on the same web site.]
E-data updates: [The department codes are updated on a monthly basis, the other codes every three months.]
Data volume: [The NHC tables comprise around 100,000 records, each around 100 characters.]

Integration responsibility:	Example solutions:	Code:
1. [The supplier must integrate the system with the NHC tables.]		

Task support: No special requirements.	Example solutions:	Code:

Data to transfer from E:	Example solutions:	Code:
2. [All codes are needed except the department codes.]		

Data recency:	Example solutions:	Code:
3. [S-data should not be older than a week.]	[The system transfers E-data every __ days. Or: IT support starts a transfer of new E-data when the Health authorities announce them.]	
3p. [Sometimes new NHC codes conflict with local codes or cause other problems.]	[IT support can roll S-data back to the previous version. Or: Local codes may have a tag so that they don't conflict.]	
4. In special cases, there may be demand for more recent data.	IT support can start a data transfer.	

Response time requirements: None.	Example solutions:	Code:

Other functions: No requirements.	Example solutions:	Code:

Data transfer to E: None.	Example solutions:	Code:

F2. Labsys

This section is an example of a close integration with an existing system. Data are transferred both ways: Requests to Labsys and replies the other way. The introduction explains what Labsys can do from a user perspective. Only task C10 uses it.

E-documentation: The customer refers to a technical document.
E-data updates: Each update corresponds to Labsys generating a reply.
S-data updates: S-data are the requests. An update corresponds to generating a request.
Data volume: A reply consists of 1000 characters on average.

F2-1 specifies that the supplier has to make the integration and may get support from the MediData company.

F2-2 says that support of task C10 must be efficient. This requirement seems a bit unnecessary since the introduction mentioned C10. However, stating it as an explicit requirement makes it easier to assess the solution. It also allows the customer to explain what he considers a good solution.

F2-3 specifies the data to transfer. The data correspond to service entities in the data model (section D3).

F2-4 and 5 specify that Labsys results must be in the EHR system (S) within 3 hours, but sometimes better recency is needed. The customer mentions a couple of solutions.

The response time requirements are already covered by L1.

F2-6 and 7 specify that the EHR system can notify its own users and Labsys about missing replies.

F2-8 specifies that the user can send Labsys requests by means of S. This is considered a kind of data transfer. It might also be called a function.

F2. [Labsys]

External system: [Labsys version yyy. Users can request lab tests from Labsys. The sample itself is delivered by … and the reply comes electronically. One reply may contain several results.]
Tasks: [Labsys is used in connection with task C10, perform clinical session.]
E-documentation: [The technical interfaces to Labsys are described in …]
E-data updates: [Labsys generates 8000 replies a day, mainly between 8:00 and 16:30.]
S-data updates: [The entire hospital generates around 8000 requests a day.]
Data volume: [Each reply consists of around 1000 characters that usually comprise several results.]

Integration responsibility:	Example solutions:	Code:
1. [The supplier must integrate the system with Labsys. MediData supports Labsys and can provide consultancy.]		

Task support:	Example solutions:	Code:
2. [The integration must support C10 in an efficient manner.]	[Requests and replies are handled in the same way as other services - without retyping patient data.]	

Data to transfer from E:	Example solutions:	Code:
3. [All E-data that can match the data in section D3.]		

Data recency:	Example solutions:	Code:
4. [S-data should not be older than 3 hours.]	[The system transfers E-data every __ hours. Or: Data is transferred at E request when they are available. Or: Data is always retrieved from E.]	
5. [The user sometimes needs the latest results for a specific patient.]	[The system retrieves data on the user's request. Or: Data is always retrieved from E.]	

Response time requirements: See L1.	Example solutions:	Code:

Other functions:	Example solutions:	Code:
6. [S can notify the user about new or missing Labsys replies.]		
7. [S can notify E about missing replies.]		

Data transfer to E:	Example solutions:	Code:
8. [The user can send Labsys requests through S.]		

F10. Integration with new external systems

Once the customer has acquired the system, it can become very expensive to integrate it with new external systems because the supplier usually has a monopoly on carrying out such changes. Section F0 (requirements 5-9) avoids the monopoly by requiring that third party is able to implement such integrations. Section F10 asks for specific features that can make it easier to do so.

The introduction explains that it is the customer's responsibility to get documentation for the external system and that he defines the update frequency, etc.

F10-1 says that the customer (or a third party) is responsible for the integration, but the supplier of the EHR system must assist him.

F10-2 specifies that the EHR system should allow an integrated system to work off-line for a period and reconnect gracefully later.

F10-3 to 6 specify features that the EHR system should provide for data synchronization when data are transferred *from* the external system: Being able to transfer data periodically or on request; asking the external system whether new data are available (the user can then decide whether to transfer it now); allow the external system to initiate a transfer to the EHR system.

F10-7 asks for a specification of the response times (transfer times). This will help the customer plan the integration solution.

F10-8 and 9 ask for a list of the functionalities the EHR system offers and a list of those it can use in an external system.

F10-10 to 12 are similar to F10-3 to 6, but specify features for data transfer *to* the external system.

F10. Integration with new external systems

As explained in F0 the customer expects that he can integrate new external systems with S. This section specifies the details.

External system: In principle any system. [Examples: X-ray system, mobile applications, specialist system for intensive care.]
Tasks: The customer's responsibility.
E-documentation: The customer's responsibility.
E-data updates: Depends on the external system.
Data volume: Depends on the external system.

Integration responsibility:	Example solutions:	Code:
1. The customer is responsible for the integration. The supplier must assist as specified in F0-9.		

Task support:	Example solutions:	Code:
2. E may in some periods be off-line, e.g. for mobile applications. In these periods, E must be able to support tasks without access to S.	S transfers new data both ways at disconnect and reconnect. Or: E is able to do it.	

Data to transfer from E: To be specified for each of the interfaces.	Example solutions:	Code:

Data recency:	Example solutions:	Code:
3. S can periodically transfer data from E.	Only new data are transferred.	
4. S can ask E whether new data of a specific kind is available [e.g. for a specific patient.].		
5. [S can on the user's request transfer E-data for a specific patient.]		
6. E can transfer data to S. Data may be any data specified in Chapter D.		

Response time requirements:	Example solutions:	Code:
7. Response times for S's technical interfaces must be specified.		

Other functions:	Example solutions:	Code:
8. [S can use functionality in E, e.g. request services or warn about missing or changed requests.]	The supplier is asked to specify the functionality S can use.	
9. [E can use functionality in S, e.g. notifying the user, or printing on printers managed by S.]	The supplier is asked to specify the functionality S provides.	

Data transfer to E:	Example solutions:	Code:
10. E can retrieve data from S. Data may be any data specified in Chapter D.		
11. S can reply to E about availability of new data.		
12. S can periodically send data to E.		

G. Technical IT architecture

The term *IT architecture* has over the years come to mean two different things. The classical meaning is the configuration of hardware, software, data communication, etc. This is the *technical architecture*. The new meaning is the technical architecture in addition to data model, usability, operation, support, etc. The template deals with this already.

Requirements to the technical architecture depend on the situation. Does the customer already have equipment that he wants to use? Or will he buy it? Or does he leave it to the supplier because the supplier is going to operate the system anyway?

The template shows an example for each of these three situations. Choose the one that fits your situation, modify it as needed, and delete the other two.

G1. Existing hardware and software

This section describes the customer's existing equipment. It also explains that other applications may run on the equipment at the same time, but they leave a certain amount of resources for the new system. Notice that free resources must be available for any 1 second period. Without this limit, the supplier cannot guarantee response times in the one-second range.

The supplier needs this information to estimate whether his system requires additional resources.

G1-1 asks the supplier to specify how many users the proposed system can support on the existing equipment. "Support" means meeting the response time, availability and storage requirements of Chapter L.

G1-2 asks the supplier to specify any additional equipment needed to reach the full number of users.

G2. New hardware and software

This section asks the supplier to specify which equipment the customer must purchase, and how it scales up according to the number of users.

G2-3 states that only equipment from the customer's favorite list should be used. This may be important if the customer has expertise in this equipment or has a purchase agreement with specific suppliers.

G3. The supplier operates the system

This section simply states that since the supplier operates the system, he decides which equipment to use.

However, there may be a need to limit the list of equipment like G2-3, because the customer wants to use his expertise to enhance or migrate the system.

G. Technical IT architecture

G1. Existing hardware and software [Alternative 1: Use what we have]

At present, the customer has the following IT equipment, which is intended for operating the new system:

1. [2 servers of type …]
2. [300 PCs with Windows XP and at least 100 GB disks.]
3. [Optical fiber net …]
4. [Oracle database …]

The equipment is used by other applications at the same time, but within these limits:

5. Within any 1 second period, servers leave 50% of the speed capacity for the EHR system.
6. [No other applications run on a PC when it runs the EHR system.]
7. Within any 1 second period, the optical fiber net leaves 50% of the capacity for the EHR system.

Platform requirements:	Example solutions:	Code:
1. Initially the system must run on the existing equipment and meet the requirements in L1, L2 and L3 for a limited number of users.	On these conditions the system can serve ___ users. [The customer expects 20 users.]	
2. In order to reach the full peak load (see L1), the system must be expanded to meet the requirements in L1, L2 and L3.	The customer has to add this equipment ____.	

G2. New hardware and software [Alternative 2: Supplier suggests]

The customer intends to buy new equipment to operate the system.

Platform requirements:	Example solutions:	Code:
1. In order to meet the requirements in L1, L2 and L3 the customer needs new IT equipment.	The customer needs this equipment _____.	
2. When the peak load grows by a factor of two, the system must be expanded to meet the requirements in L1, L2 and L3.	The customer has to add this equipment ____.	
3. As far as possible, only equipment from the list in appendix X should be used.		

G3. The supplier operates the system [Alternative 3: His problem]

Platform requirements:	Example solutions:	Code:
1. The supplier operates the system and uses the necessary equipment to meet L1, L2 and L3.		

H. Security

Security comprises many areas: checking the user's access rights, giving them the necessary rights, protecting against data loss, protecting against other threats.

H1. Login and access rights for users

This section describes the situations where the user's access rights must be checked. The requirements are expressed as subtasks to be supported and problems to be removed. The template shows two alternatives: (1) The new system must do as our other systems. (2) The new system should provide better security.

Alternative 1: Login as today

Requirement 1 says that the user must be identified. The example solution is to use the existing approach.

Requirement 2 says that access is only allowed to users with the proper rights. The example solution mentions two ways to do it.

The list of access rights is shown below the requirements table. A crucial point is the granularity of the rights. Does the user get the right to prescribe medicine in general or only medicine in a specific department? In the example, the granularity is a department.

Many customers neglect this list although it is important for the supplier's assessment of the solution complexity. Assigning the proper rights to the users is not technically difficult, but checking the rights with the proper granularity is often complex and has to be handled deep down in the system.

Alternative 2: Better login wanted

Requirement 1 again says that the user must be identified, but the example solution calls for an alternative identification.

Requirement 2 asks for support of the situation where user 1 has been away from the system for some time and another user may access the system with user 1's rights. The traditional solution is time out, but it causes problems that need support.

Requirement 3 says that the rights must be checked and mentions the existing problem with a password for each system. One solution is mentioned: single sign-on. (This is only part of a solution because the customer's other applications must be changed to follow the same scheme. This is not the EHR supplier's responsibility.)

H. Security

H1. Login and access rights for users

Login is not a separate user task, but subtasks that occur in many tasks. The system must support the following subtasks relating to the user's access rights.

[Alternative 1: Login as today]

Subtasks for user access rights:	Example solutions:	Code:
1. Identify the user	The system uses the existing user identification, login method, and time-out method, which is …	
2. Check that only authorized users get access to systems and data. (See the requirement note below.)	The database system checks the rights. Or: The user screens show only the authorized functions and data.	

[Alternative 2: Better login wanted]

Subtasks for user access rights:	Example solutions:	Code:
1. Identify the user.	A user identifies himself with a user name and a password; preferably also an alternative identification such as voice or finger print recognition.	
2. The user has been away from the system for some time.		
2p. Problem: Another user may access the system with the rights of the first user.	The system times out after 10 minutes of non-use.	
2q. Problem: If the system logs out automatically, it is cumbersome to log on again.	The system requires password only. The timeout period may depend on the physical location, for instance a long timeout in the operating room.	
2r. Problem: If the system logs out automatically, entered data may be lost.		
3. Check that only authorized users get access to system functions and data. (See the requirement note below.)	The database system checks the rights. Or: The user screens show only the authorized functions and data.	
3p. Problem: Today the users have a password for each system. It is cumbersome to switch system and hard to change passwords regularly. As a result, users tend to post passwords where everyone can see them.	Each user has only one user name and one password (single sign-on).	

Requirement note: Granularity of access rights

1. [Right to prescribe drugs in department M.]
2. [Right to see patient data in department M.]
3. [Right to record clinical data (diagnoses and services) in department M.]

…

[A surgeon in department M might for instance have rights 1, 2, and 3, while a supervising surgeon for department M has rights 2 and 3 only.]

H2. Security management

Security management assigns and removes user rights, defines new roles, etc. An organization may have central security management or delegate it to departments. The example specifies that it must be local.

The template describes security management as subtasks to be supported and problems to be removed. One of the problems is to assign rights to many users when they start working at the beginning of the month.

Some of the solutions are well-known techniques such as role-based rights and time-limited rights.

H2. Security management

Security should be managed locally.
The work in security management includes the following subtasks.

Subtasks for security management:	Example solutions:	Code:
1. A central department will assign and remove rights for all users. See 2q for an exception.		
2. Assign or remove access rights for a user.		
2a. Create the user.		
2p. Problem: A lot of users need access rights when they start the first day in the month.	The system transfers data from the personnel system once a month.	
2q. Problem: A temporary employee has been appointed in a hurry, but is not yet in the personnel system. Needs access rights anyway.	Possibility for temporary registration in the department, bypassing the central department.	
2r. [Problem: Security management must keep track of the relationship between 4000 users and 300 rights.]	[Each user is assigned one or more roles, e.g. surgeon in department M and supervising in department N. Each role has one or more rights, e.g. prescription and clinical recording.]	
2s. Problem: Security management forgets to assign and remove rights on the right dates, e.g. in connection with hiring and resigning.	Rights and roles can be defined ahead of time and be valid for a certain period, e.g. from the day the person is employed.	
3. Create new roles with new combinations of rights.		
4. Get an overview of who has which rights and whether some rights have not been assigned to anyone.		

H3. Protection against data loss

The template mentions some typical risks of losing data, and the supplier is asked to describe his solution. For disk crashes and fire, the template suggests the traditional solutions.

With the help of a security expert, the customer may ask for protection against many other sources of data loss.

H4. Protection against unintended user actions

This section mentions typical risks caused by users unintentionally doing something unexpected.

Requirement 1 says that no user action may cause the system to close down. This is a tacit requirement to all systems and if not written it might still hold in court. Writing it, however, removes any doubt. The example solution mentions how the customer could be convinced.

H3. Protection against data loss

Data may unintentionally be lost.

The system must protect against:	Example solutions:	Code:
1. Disk crash	Periodic backup or RAID disks.	
2. Fire	Remote backup …	
3. (See F0 for protection of data against loss or replication during transfer between systems.)		

H4. Protection against unintended user actions

An unintended user action means that the user happened to do something he didn't intend to do, e.g. hitting the wrong key or using a command that does something he didn't expect.

Requirements:	Example solutions:	Code:
1. Unintended user actions may not cause the system to close down, neither on the client nor on the server.	The supplier's issue log and a description of the supplier's test methods might provide evidence.	
2. All data entered must be checked for format, consistency and validity. In case of doubt, the user must be warned and asked what to do.		
3. The user must be able to correct mistakes easily.	The system provides extensive use of undo.	
4. The user must be able to interrupt long functions, e.g. waiting for a remote data transfer.		

H5. Protection against threats

This section deals with threats caused by viruses, hacking, etc. In order to identify the most important ones, you carry out a security risk assessment.

You look at the potential threats one by one, estimate the frequency of their occurrence and the consequence when they occur (preferably in money terms). Then you calculate the "average" damage per year for each threat. Based on this, you select the most serious threats.

Alternative 1: The customer knows the risks

The customer has made a security risk assessment and has listed the serious threats. He then asks the supplier to suggest a protection. The template shows only a few examples.

We often see security requirements that specify a solution more than a need. As an example, we see requirements like this:

> *The password must be at least 9 characters with at least one capital letter*.

This is cumbersome to the user, so let us ask the security specialist why this is necessary. *Well*, he says, *an intruder might try all possible passwords with a special program. If the system handles login attempts at full speed, it is just possible to break eight-character passwords in a reasonable time*.

Requirement 3 handles this as a threat. We can now see that there are other solutions. The solution column mentions two that are far more convenient.

Requirement 5, *preventing unauthorized persons from accessing personal data*, sounds easy, but it comprises a lot of independent threats, such as wire tapping and IT staff looking at the data on the disk. The supplier's proposal can easily become a long novel - and it is hard to compare two suppliers' novels. We suggest omitting this requirement and ensuring that the risk assessment covers the threats and includes the serious ones as requirements in H5.

Requirement 6 tries to solve the problem by referring to a law on the matter. This is fine because laws must be followed. However, it often creates an interesting game between customer and supplier. The customer hasn't read the law in question, but imagines that it covers the threats (it only partly does so). He reasons that if he requires the supplier to follow the law, then the supplier has the responsibility for adequate protection.

Most likely, the supplier knows the law and knows that it doesn't cover adequately. He also knows that the purpose of the customer's requirement is to renounce the responsibility, and that the law will not be verified at delivery time. Why should he point this out to the customer? The result is that the real protection demand isn't covered.

We suggest that requirement 9 is considered an addition to the security risk assessment and the specific threat requirements (1-4 in the example).

Alternative 2: No risk analysis has been made

There is only one requirement: The supplier is asked to list the important risks and propose safeguards. Notice that we don't ask him to make a risk assessment but only list typical threats for this kind of project. If we talk about simple applications such as web shops, and the supplier has expertise in the area, this is sufficient.

However, in unusual projects the customer should ask the supplier to make a specific assessment with customer data. This is costly to both parties, so it should be made during the project, maybe during the early proof-of-concept.

H5. Protection against threats

[Alternative 1:] A risk assessment has shown that the following threats are the most serious. The system must protect against them.

The system must protect against:	Example solutions:	Code:
1. Unauthorized persons obtaining manager rights through the internet (hacking).	The rights can only be used on the internal network.	
2. Wire-tapping of passwords.	Password encryption.	
3. An intruder tries all possible passwords with a special program.	Passwords must be at least 9 characters (cumbersome). Or: 5 second delay between login attempts. Or: Access blocked after 3 attempts.	
4. SQL injection (makes the system handle data entry as a database command).		
5. Unauthorized persons getting access to personal data. [Too open-ended, see the guide booklet.]		
6. [The system must conform to Law on Handling of Personal Data (Law 429, May 31, 2000).] [Okay, but check for completeness. See the guide booklet.]		

[Alternative 2:] The customer has not made a security risk assessment.

Threat protection:	Example solutions:	Code:
1. The supplier lists the threats that are most serious for this kind of project and specifies the safeguards he proposes.		

I. Usability and design

Usability means that the system is easy to learn, efficient for the frequent user, easy to remember for occasional users, easy to understand - also in unusual situations, and pleasant to use. These *usability factors* are not equally important. Importance depends on the kind of system we specify.

When talking about usability, we assume that the system from a technical viewpoint works correctly and replies fast, and that it actually can support the tasks. Nevertheless the users have troubles using the system.

Many developers and designers believe they can see whether the system has adequate usability. Or that it is good enough if an expert user says so. It has been proven over and over that this is not possible. Usability has to be tested and measured with real, potential users.

Usability can be measured in many ways. The most important is to observe users carry out some realistic tasks by means of the system or a primitive prototype of it. We log when the user gets stuck and needs help, when the user spends too much time finding the solution, etc. This is called a *usability test*. The problems we log are called *usability problems*.

We can rather objectively classify the problems as critical or less critical. The template shows how as a requirement note. We may then express the usability requirements as the allowed number of critical problems. Notice that a problem is critical only when two or more users have experienced it. The reason is that a large number of usability problems are only observed once (*singular problems*). Usually it doesn't pay to try removing them.

We may ask the user to *think aloud* during his attempts. This gives us far better possibilities for understanding why the user encountered the problems, and the developers get a better chance of removing the problems.

Experience shows that usability problems must be detected and removed before programming. Later on it is too expensive to remove the many problems that require program changes. To achieve this, we draw mockups of the screens with paper and pencil or simple computer tools. We use the mockups for think-aloud usability tests. Most usability problems can actually be detected this way. Next we modify the mockups to remove the problems, and test again. This approach is the basis for the early proof of usability in B2-2.

I1. Ease-of-learning and task efficiency

This section of the template states the usability requirements in such a way that we during the early proof-of-concept can estimate whether the system will get sufficient usability. At the same time we design a detailed user interface. Experience shows that development is faster when a detailed, proven user interface is known early on.

Before the proof of concept it may be hard to specify the exact way of measuring the usability, and the customer may easily state unrealistic usability requirements. As an example, imagine that we deleted requirement 1 in the template and kept requirements 2 to 6. We would thus require that users were able to carry out all

I. Usability and design

I1. Ease-of-learning and task efficiency

It is important that the system obtains adequate usability. This is best done through early usability tests. After the early tests, customer and supplier jointly decide the detailed requirements to be verified at the time of system delivery. This may be a modified version of the requirements below and a detailed specification of the test tasks.

[If the parties cannot agree on the detailed requirements, they may cancel the contract (cf. section B2-2).]

Requirements for early proof of concept:	Example solutions:	Code:
1. The parties must test the user interface for usability soon after signing the contract. The critical usability problems must be corrected until usability testing gives acceptable results (see the requirement note below). In addition the parties must agree on the detailed usability requirements.	Usability testing (think-aloud testing) is carried out for existing parts of the system in a suitable setup. For parts that don't exist yet, think-aloud testing is done with paper mockups. Three new users participate in each round of testing.	

Requirements to be agreed in detail during the early proof of concept, and verified at the time of delivery:		
2. After a short instruction by super users, the ordinary users must be able to carry out all tasks in Chapter C within their own work areas with few critical usability problems.	Within each work area, thinking-aloud testing is done with three randomly selected users. A maximum of ___ critical usability problems may be observed.	
3. Error messages must be understandable and helpful.	During the usability test, a selection of error messages is shown to the user, who tries to explain what the message means and what to do about it. __% of the explanations must be acceptable.	
4. It must be possible to operate the system with keyboard only. Users must be able to learn it on their own.	Late in the usability test, the user is asked to use keyboard only. __% of the users must be able to do so.	
5. Super users must be able to learn the system quickly so they can train other users (cf. J2-1).	Training of a super user takes ___ days. (The customer expects 3 days).	
6. [A user who has used the system for a week, must be able to quickly order a set of services for a patient, e.g. lab test, scanning …]	[A typical user is able to order these services in __ minutes.]	

Requirement note: Serious and critical usability problem
A **serious** usability problem is a situation where the user:
a. is unable to complete the task on his own,
b. or believes it is completed when it is not,
c. or complains that *it is really cumbersome*,
d. or the test facilitator observes that the user doesn't use the system efficiently.

A **critical** usability problem is a serious usability problem that is observed for more than one user.

tasks with few critical usability problems, were able to understand error messages, etc.

In his proposal, the supplier would have to specify the allowed number of usability problems, misunderstandings, etc. This is close to impossible for system parts that don't exist yet. One purpose of requirement 1 is to find some reasonable usability requirements early in the project.

Requirements 2 to 6 are outlines of usability requirements that have to be finished during the proof of concept. The number in column 2, for instance, has to be filled in.

Requirement 2 checks that after the planned introduction, users can carry out their tasks with minimal support from others.

Requirement 3 checks that error messages are usable. During usability tests, the users will only encounter a few error messages. Requirement 3 makes it possible to test more messages, also those that rarely occur.

Requirement 4 says that it must be possible to operate the system without a mouse, and users must learn it on their own. For some systems this may not be relevant, for instance for applications used occasionally.

Requirement 5 deals with large systems that typical users cannot learn on their own. Traditionally, customers ask for courses that all users must take, but it is often an expensive and inefficient approach. Instead we ask for ways the super-users can learn the system and train other users. One way is to provide courses for them. In J2-1 we ask the supplier to run such courses.

Requirement 6 deals with efficiency for the frequent user. During the early usability tests, we may get a feeling for how fast users should be able to work, but we cannot measure it until the system is operational.

Web systems

The template shows requirements suited for production systems that are used on a daily basis. Requirements 2, 5 and 6, for instance, are not suited for websites used occasionally. There are no super users around, and efficiency is unimportant. However, requirement 4 may still be important if the site is intended for visually impaired users. They have troubles hitting with the mouse.

I2. Accessibility and Look-and-Feel

Some usability aspects are hard to express through usability tests. Rules and standards may be better.

Requirement 1 says that the user interface must follow the MS-Windows guidelines. Notice that the reason is stated: most users are familiar with Windows, and the guidelines will make the system easier to learn. If you don't have a good reason, there is no need to follow a guideline. Many people believe that a guideline ensures usability. It does not. At most it contributes a bit, and in some cases it may even be harmful. Following a guideline is not free. It is amazingly difficult to check that the guideline is followed - and correct the mistakes.

Requirement 2 says that the user interface must be suited for blind and visually impaired users. One solution is to follow the HTML principles, which were developed for this purpose. As an example, standard heading tags should be used rather than self-defined, visually impressive styles. Heading tags allow screen reader programs to use intonation for "highlighting" the headings. In the same way, fixed column widths and font sizes should be avoided so that visually impaired users can enlarge the text many times.

I2. Accessibility and Look-and-Feel

Requirements:	Example solutions:	Code:
1. The user interface must follow the MS-Windows guidelines, which most users are familiar with.		
2. Web pages must be suited for screen readers, scaling to visually-impaired users, and utilizing the full screen size on small as well as large screens.	The pages follow the HTML guidelines for Accessibility (WCAG10 from W3C).	
3. [The user interface must be in Danish. The pages with opening hours, phone numbers, and addresses must be available in Danish, English, Turkish, and Urdu.]		

Some requirements specifications replace requirement 2 with a requirement that the web pages must pass a W3C Markup validation test (http://validator.w3.org). This test analyzes the web pages and finds errors. This is yet another example of analysts prescribing a standard in the belief that it covers the demands. The test only finds formal errors, for instance missing end tags or missing quotes. It doesn't say anything about suitability for the blind. The guidelines in WCAG10, however, have rules for supporting the blind.

Requirement 3 is an example where the language must be specified.

J. Other requirements and deliverables

This chapter collects requirements that don't fit into the other chapters. The template shows only some of the areas that belong to this chapter.

J1. Other standards to obey

Most required standards belong to other chapters, for instance data and security. The rest may be stated here.

In practice we see customers write a long list of standards, often without knowing what they cover. Usually it is cumbersome to check whether a standard is met. As a result a careful supplier must increase the price, while a less careful supplier assumes that the customer doesn't check whether the standards are met. (See the examples in H5 and I2.)

The template shows only a single example of a standard (of the soft kind). The supplier is required to obtain the certification, i.e. an independent check that the system meets the standard. This relieves the customer of the need to check for himself.

J2. User training

User training is often forgotten - or an unrealistic amount of training is requested. Often the training takes place at the wrong point in time, for instance so early that users have forgotten all of it when the product finally arrives.

Requirement 1 is an example where the customer realizes that only super users need training from the supplier. We ask the supplier to train 50 super users. The training must enable them to train other users. This is in recognition of the fact that most supplier courses are too far from the user's real tasks. The idea is to use super users as mediators. It is specified what the super users must be able to do after the training (see also I1-5).

Requirement 2 specifies similar requirements for training the customer's IT staff.

Requirement 3 specifies when the training must take place relative to system delivery.

J. Other requirements and deliverables

J1. Other standards to obey

Requirements:	Example solutions:	Code:
1. [The system must follow good accounting practice. The supplier must obtain the necessary certification.]		
2. …		

J2. User training

The customer wants to deliver a large part of the training himself. The idea is to train super users first and then let them train others.

Requirements:	Example solutions:	Code:
1. [The supplier must train 50 super users, making them able to train other users. The training must enable the super users to carry out all tasks in Chapter C, including variants, within their own work areas.]	Training of a super user takes __ days. (The customer expects 3 days).	
2. [The supplier must train 10 IT staff, making them able to handle the customer's part of system operation and support.]	Training of IT staff takes ___ days. (The customer expects 10 days).	
3. The training must be carried out within the last month before system delivery in order that users and IT staff can use the system immediately and haven't forgotten what they learned. If necessary, the training must be repeated and the delivery delayed.		
4. …		

J3. Documentation

User and system documentation are often forgotten too. The example points out that full documentation isn't needed for everybody. This is in recognition of the fact that few users read the documentation or on-line help, even if it is available and reasonably useful. This recognition may save many expenses and frustrations for both parties.

Requirements 1 and 5 specify that course material must be available to super users when they train other users, i.e. before system delivery. It must be available in a form that allows the super users to adapt it, for instance with examples from the customer's world. Requirement 2 specifies that full documentation for super users must be available shortly after system delivery.

Requirement 3 specifies in the same way the documentation for the customer's IT staff.

Requirement 4 specifies documentation for specially developed software and technical interfaces. The criterion is that the documentation must be sufficient for third party to maintain these parts and to transfer data to another system. To ensure that the supplier can actually deliver the necessary documentation quality, you can ask for an early proof as in section B2.

J4. Data conversion

Data conversion from previous systems to the new system often makes up a significant part of the supplier's price. This section specifies what to convert. It is important that the customer documents the data formats since the supplier must otherwise obtain the information from other sources in order to calculate the correct cost. This may scare good suppliers from bidding.

Validation of the conversion is a large problem that some suppliers know much more about than the customer. For this reason, requirement J4-3 asks the supplier to explain how he will do the validation.

J5. Installation

This section specifies who installs what. If the customer wants to install the system himself, he may ask for the necessary documentation and an estimate of the time it will take.

J3. Documentation

The customer expects that only super users, IT support staff, and systems developers will read the documentation. Thus there is no need for beginner's documentation, except for course material.

Requirements:	Example solutions:	Code:
1. Before system delivery, course material must be available for super users to use when teaching other users. (The customer contributes with documentation of the future work processes, see K-10.)		
2. A month after system delivery, user-oriented documentation of all system functions must be available. The documentation must be suited for super users.		
3. Before system delivery, sufficient documentation must be available for the customer to handle his part of IT operation and support.		
4. For specially developed software and technical interfaces, sufficient documentation for third-party development must be available two months after system delivery.		
5. All documentation must be delivered in electronic form. The customer may freely modify it and copy it for his own use.		
6. …		

J4. Data conversion

The supplier must convert the following data from the existing systems:	Example solutions:	Code:
1. [Those data from the patient management system that the EHR system will handle in the future. The format is described in …]		
2. [Those data from the XY system that the EHR system will handle in the future. Data must be transferred through IBM 3270 emulation. See the screen format in …]		
3. All converted data must be validated.	The supplier is asked to describe how.	
4. …		

J5. Installation

Requirements:	Example solutions:	Code:
1. The supplier must install all parts of the delivery, hardware as well as software.		
2. The supplier must install all converted data.		
3. …		

K. The customer's deliverables

Most of the requirements specify what the supplier must deliver. However, an IT system isn't something that the supplier just rolls in and plugs into the power outlet. The customer's employees have to contribute in various ways, and the supplier's employees may need office space and other facilities during development and deployment.

This chapter specifies what the customer has to provide. The supplier may in column 2 specify what he expects, and in his proposal he may add new points to the list.

In many contracts, this chapter is replaced by a separate contract appendix.

Like other sections of the template, the requirements in this chapter are only examples and not an exhaustive list. So take care: In many countries legal practice is that the contract must specify everything the customer has to deliver. After signing the contract, the supplier cannot expect office facilities or expertise in some customer area unless it is specified in the contract or its appendices.

K. The customer's deliverables

The following list of the customer's deliverables and services must be complete. The supplier cannot expect more from the customer. If necessary, the supplier must add to the list in his proposal.

The customer delivers:	Example solutions:	Code:
1. [Hardware, software, and external systems that the EHR system requires (see the details in Chapter G). The equipment must be available when the installation test starts.]		N/A
2. [Office with three IT work places from one month before the planned installation test to one month after system delivery.]		N/A
3. [Samples of production data for testing purposes and the full data set for conversion, except data from the XY system.]		N/A
4. Test cases for deployment testing.		N/A
5. Expertise in the application area corresponding to a half-time employee during the entire project.		N/A
6. Test subjects for usability tests.		N/A
7. A half-time project manager and a half-time secretary.		N/A
8. Super users/instructors who learn the system in order to train ordinary users.		N/A
9. Expertise for validation of converted data.		N/A
10. Contribution to the course material on future work processes (cf. J3-1).		N/A

L. Operation, support, and maintenance

This chapter specifies the supplier's responsibilities after delivery of the system itself. These requirements can only partly be verified (tested) at the deployment test. We may for instance set up a simulation of 2000 users and measure response times, or we may test that the support organization works, but we cannot test that it also works well when 2000 real people work with the system.

The full verification takes place after delivery, for instance at the operational test or through investigation of logs and statistics.

The template corresponds to the situation where the supplier is responsible for operation, support, and maintenance. If the supplier for instance isn't responsible for support, the corresponding section should be empty. In this case the customer may need courses and documentation that allows him to support the system. Requirements for this are stated in Chapter J.

If the supplier isn't responsible for operations, we cannot just delete sections L1 (response times) and L2 (availability). The supplier is still responsible for the response time - assuming that the system runs on the configuration described in Chapter G. Similarly the supplier is responsible for part of the availability. If the system breaks down due to errors in his software, he is responsible for the corresponding lack of availability. This is explicitly stated in section L2.

In many contracts, this chapter is moved to separate contract appendices.

L1. Response times

The introduction part describes the estimated load of the system in the busiest periods (*peak load*). Without these figures, the supplier cannot estimate the response times and the necessary hardware. It may be necessary to specify more on the load, for instance if other response times are acceptable or needed in other periods or for other user groups.

The template shows two different ways of expressing the load:

1. The number of users who carry out the various tasks. If the supplier has in-depth knowledge of the work areas, he can translate the figures to the number of IT transactions, which in turn allows him to estimate the response times. Without this knowledge, it is very risky to promise response times.

2. The number of IT transactions. When the supplier is not expected to be familiar with the work area, the customer should specify the number of IT transactions rather than the number of users. Even if the customer doesn't know the number, the requirements should still be expressed as the expected number of transactions since the customer ought to know more about it than the supplier.

The template mentions that the customer has estimated the number of transactions from the task frequencies stated in Chapter C and the yearly data volumes in Chapter D. In order to translate this to transactions in peak load, he also needs to know the percentage of transactions made in peak periods and how many transactions an average task will generate.

L. Operation, support, and maintenance

This chapter specifies the supplier's responsibilities after delivery of the system itself. These requirements can only partly be verified (tested) at the deployment test. The full verification takes place later, at the operational test. [Depending on the supplier's responsibility, only some of the requirements are relevant.]

L1. Response times

It is important that response is so fast that users are not delayed. Response time is particularly important during the busiest hours, the **peak load** periods, which are [morning 9-11 and …]

Peak load [Alternative 1: Only suited when the supplier knows the domain intimately**:**

1. 2000 users work with clinical sessions (C10).
2. 1000 users work with patient management (C1 to C4).
3. 300 users browse the public web part.]

Peak load [Alternative 2: Suited when the supplier doesn't know the domain intimately:
The figures are estimated from task frequency (Chapter C) and data volumes (Chapter D).

1. Simple queries in clinical sessions (C10): 10 per second.
2. Updates in clinical sessions (C10): 2 per second.
3. Public web access: 5 page loads per second.]
4. …

Measuring response time
The response time is the period from the user sends his command to the result is visible and the user can send a new command. A command means a key press or a mouse click. All measurements are made in peak load periods with the actual number of users, assuming that the load is within the limits above.

Fractile: The times specified below must apply for 95% of the measurements.

Production work: Measurements are made with a setup according to Chapter G.

The public web part: Measurements are made on a PC connected to the Internet through a 56 KB modem with low traffic on the route to the servers, but with peak load of the servers themselves.

The introduction part specifies how the response time is to be measured and under which circumstances. In the example it is done differently for the production work on the local-area network and for the public web part. Many users of the public web part still have a slow connection, and the supplier should avoid too many pictures, which would slow down the page presentation. The fractile of response times that must be within the required limits is also stated (more on this below).

Requirement 1 simply says that there is a need to measure regularly - and in the peak load periods. In column 2 the customer has given examples of how it might be done. The supplier will specify his solution according to what is feasible for him.

Requirements 2-9 specify the required response times. They are based on ergonomic measurements of how people work at computers (the *keystroke-level model,* Card et al., 1980). A fast user types 5-10 characters per second, so 0.2 seconds to move from one field to the next on the screen, will barely slow down the work.

A user spends around 1.3 seconds to change focus from one "mental chunk" to another, for instance from entering client data to entering the client's request. If the screens are structured accordingly, a 1.3 second screen switch will not slow down the user. This applies to requirements 3, 4 and 5.

In practice there will be cases where the system needs more time to reply, and where the user expects it. Here we meet an ergonomic constant of 20 seconds. Even when the user knows that it takes time, he will unconsciously wait around 20 seconds and then start working on something else. Switching from one task to another takes time - wasted time. For complex tasks the mental switch time might be as long as 10-20 minutes. Requirements 6 and 7 are examples of occasionally used functions where 20 seconds are acceptable.

Finally there may be functions where we for technical reasons expect response times above the ideal. Requirements 8 and 9 (login) are examples of this. Ideally, login should take place within 1.3 seconds.

The supplier may in column 2 specify functions that don't follow the common response time rules, for instance an overview screen that may take 3 minutes to display.

Web systems

The response times in the example are for production work through a local area network. For websites used occasionally, these requirements are much too strong and meeting them would be unnecessary and costly.

Fractile: valid in X% of the cases

Notice that the required response times must be valid for only 95% of the cases (or some other fraction). Why not ask for 100% of the cases? Because it is unrealistic in a multi-user system.

Let us look at an example: The system takes 0.2 seconds to handle a request. When a request arrives while the system is idle, its response time will thus be 0.2 seconds. Assume that there on *average* are 2 requests per second. This means that the system will be busy 40% of the time. If another request arrives while the system is busy, it will be queued and its response time becomes larger than 0.2 seconds. This will happen to 40% of the requests.

The result is that 95% of the requests get an answer within 0.6 seconds, and 99% get an answer within 0.8 seconds (for an M/D/1 queuing model).

(L1. Response time, continued)

Response time requirements:	Example solutions:	Code:
1. Response time measurements must be made regularly in the peak load periods.	Measurements are made once a week with a stop watch. Or: The system measures all the time.	
2. When moving from one field to the next, the user's typing speed must not be slowed down.	Typing is possible within ___ s. (The customer expects 0.2 s.)	
3. When moving from one screen to the next, data must be visible and typing possible within the mental switching time.	Data is visible and typing possible within ___ s. (The customer expects 1.3 s.)	
4. Lookup in drop-down lists must allow selection from the list within the mental switching time.	Selection is possible within ___ s. (The customer expects 1.3 s.)	
5. Reports used frequently must be visible within the mental switching time.	The report must be visible within ___ s. (The customer expects 1.3 s.)	
6. Reports used occasionally must be visible before the user loses patience.	The report must be visible within ___ s. (The customer expects 20 s.)	
7. Data from external systems must be transferred and displayed before the user loses patience. For some time after this, the data must be accessible as fast as other data.	Data is visible within ___ s. (The customer expects 20 s.) Data may for instance be cached locally for some time.	
8. Login must be completed before the user loses patience.	The user can start working within ___ s in addition to the time he spends typing his identification. (The customer expects 10 s.)	
9. Repeated login when the user temporarily has left the system, must be completed before the user loses patience.	The user can start working within ___ s in addition to the time he spends typing his identification. (The customer expects 4 s.)	

This isn't scary, but what about worst case? There are on average 2 requests per second, but by coincidence - or because we test this way - 2000 users may send a request almost at the same time. The system processes 5 requests per second and as a result the last user gets a reply after 400 seconds.

If we had asked for a maximum response time of 2 seconds - also in this extreme case - we would need a system that could process 1000 requests per second. It would be extremely expensive. So never ask for a maximum response time in a system with many users.

L2. Availability

Availability is the fraction of time where the system must be operational from the user's perspective. We have to define more precisely what it means that the system is out of operation, and how we deal with cases where some users can access the system but others cannot. If only one user cannot access the system, we would hardly call it a system breakdown.

A breakdown can have many causes and the template mentions 5. Not all of them are the supplier's responsibility. When the supplier isn't responsible for operation, he will still be responsible for breakdowns with cause 3 (errors in software or configuration). When the supplier is responsible for the operation, also power failure, hardware breakdown, capacity problems, etc. are his responsibility.

In principle the customer can state all kinds of requirements for calculating the availability, but in practice he must accept the possibilities the supplier can offer - as long as they cover his real needs.

The introduction part suggests one way to calculate a breakdown period: A breakdown is always calculated as 20 minutes. An operational period must last at least 60 minutes. The reason is that users don't resume their interrupted tasks until around 20 minutes after the breakdown, and they cannot produce much in an operational period less than an hour.

The template also suggests a way to calculate the availability when only some of the users are affected by the breakdown.

Notice requirement 1, which says that the availability must be calculated periodically. This means that excess availability cannot be transferred from one period to the next. In column 2 the customer has suggested that availability is calculated as described in the introduction part. The supplier may propose his own way of calculating the availability, for instance by referring to an appendix.

Requirements 2-3 state the required availability in two different operational periods. Take care not to ask for too much. It may be very expensive. Section A2.3 of the template (example L2) shows how the customer could lose a lot of money by being too demanding. An availability of 99.5% means that the system may be out of operation 8 hours a year in normal work hours. An availability of 99.9% means 1.6 hours a year, but for a large system the cost may be several million dollars more.

L3. Data storage

This section specifies the amount of data to be stored. The example distinguishes between data with immediate access and archived data with slower access. Certain kinds of pictures are stored for a shorter time.

The example refers to the detailed data volumes in Chapter D, where each table has a total size and sometimes a yearly growth. We might also specify all table sizes here in section L3 and remove them from Chapter D. Keeping them in both places would be convenient, but easily creates inconsistencies.

L2. Availability

The system is out of operation when it doesn't support some of the users as usual. The cause of the breakdown may be:
1. The customer's issues, e.g. errors in the customer's equipment.
2. External errors, e.g. power failure.
3. The supplier's issues, e.g. errors in software or configuration.
4. Planned maintenance.
5. Insufficient hardware capacity.

Solution note: Measuring availability
A breakdown is counted as at least 20 minutes, even if normal operation is resumed before. If the following period of normal operation is less than 60 minutes, it is considered part of the breakdown period.

[When the supplier is not responsible for operations, only breakdowns with cause 3 are included in the availability statements. When the supplier is responsible for operations too, he may also be responsible for causes 2, 4, and 5.]

The **operational time** in a period is calculated as the total length of the period minus the total length of the breakdowns for which the supplier is responsible. The **availability** is calculated as the operational time divided by the total length of the period. When only some of the users experience a breakdown, the availability may be adjusted. One way is to calculate the availability for each user and take the average for all users.

Availability requirements:	Example solutions:	Code:
1. The availability must be calculated periodically. The calculation should compensate for the number of users experiencing breakdowns.	The availability is stated monthly and calculated as described above.	
2. [In the period from 8:00 to 18:00 on weekdays, the system must have high availability.]	In these periods the total availability is at least ___%. (The customer expects 99.5%)	
3. [In other periods the availability may be lower.]	In these periods the total availability is at least ___%. (The customer expects 98%)	

L3. Data storage

The data volume is specified in Chapter D. Data must be stored as follows:

Data storage requirements:	Example solutions:	Code:
1. [The system must give access to data for the last 5 years with the response times specified in L1. This includes the history trail.]		
2. [X-ray pictures are only kept for … years.]		
3. [The system must give access to archived data for the last 20 years with response times as for external systems (L1-7). History trails are not needed.]		

L4. Support

This section specifies the supplier's support services, for instance helping users (hotline), changing the system configuration, and monitoring operations. (ITIL has specific terms for this. Hotline is for instance called *Service Desk.* See Bon, 2004.)

The introduction states that super users are the first point of contact. If they cannot remedy the problem, the super user or the ordinary user may contact hotline. We might allow ordinary users to contact hotline directly, but in most organizations it would be much more expensive, and less effective.

The introductory requirement note explains what it means to handle a request for help (an *incident* in ITIL terminology). It is described as a list of subtasks. After most of the subtasks, the user gets a first reply. A reply means that the user has got help in solving or circumventing the problem, or that a technical problem has been remedied, or that the problem has been transferred to another organization. It is not a valid reply that the request has been received by the hotline or transferred to another *supporter*. The user may often get a first reply and later additional replies as supporters investigate the case.

Requirements 2 and 3 specify when users can contact the hotline by phone or in person (direct contact), and that the supporter must try to resolve the problem on the spot.

Requirement 4 specifies that for indirect contacts the user must get a first reply within a few hours.

Requirements 2 and 4 specify the speed of the service. Similar to the response time requirements, the service speed must be valid in 95% of the cases. Don't specify a maximal time for a reply (valid for 100%). The worst case, where everybody asks for help at the same time, will be excessively expensive to handle.

Requirement 3 asks for on-the-spot handling of direct contacts. Many SLA's (Service Level Agreements) specify that a certain fraction of the requests must be resolved on the spot. Experience shows that this makes the supplier interested in getting a lot of trivial requests. He is not motivated to prevent them, for instance by broadcasting how certain problems can be avoided.

For this reason requirement 3 only asks the supporter to spend a few minutes on the spot. Whether the support quality is adequate in general is hard to measure. Requirement 10 suggests that the parties discuss this at regular meetings.

The example asks for specific services such as remote diagnostics and sending a support person to the customer's site. As for other requirements, the supplier may respond that he doesn't provide this.

Like other sections of the template, the support requirements are only examples and not an exhaustive list. The ITIL specifications may be used for creating a longer list of support processes. As with other standards, don't just use them blindly. You may end up paying for more than you need or asking for inconvenient processes, such as *always send the reply back to the user through the first point of contact*.

L4. Support

Support comprises help to users, configuration changes, and monitoring of the operation. In this chapter, "supplier" means the supplier's IT and support organization. A "supporter" means a qualified person from the supplier. The specified response times must apply for [95% of the cases].

[Super users are the ordinary user's first point of contact. The supplier only has to help when the super users cannot remedy the problem. The supplier may interact with the super user or directly with the user.]

Requirement note: Handle a request
Whenever a supporter receives a request, he can perform one or more of the following subtasks. All subtasks except e (escalation) end with a **reply** to the user. The request is **closed** when nothing more can be done about the request (subtask f).

a. Help user: Assist the user in solving the problem or circumventing it. If needed contact the user for clarification. Assistance is considered a valid reply.
b. Change configuration: E.g. start servers, change settings, replace printer cartridges, install software. Reply to the user when it has been done.
c. Order equipment or help from another organization: Reply to the user about the expected delay.
d. Defect: The support organization cannot solve the problem. Report it to the maintenance organization. Reply to the user that it has been done.
e. Escalate request: The supporter cannot fully solve the problem himself. Pass the request on to another supporter. This person may again perform one or more of the subtasks.
f. Close the request: Nothing more can be done about the request. This may happen at the first point of contact. The request may also escalate several times, wait for external delivery or wait for a reply from maintenance before it can be closed. Reply to the user that the request has been closed.

Support requirements:	Example solutions:	Code:
1. The supplier must handle user requests for help. Help must cover all equipment and software provided under this contract.		
1p. Problem: Even super users cannot decide which product a specific problem relates to. It is even harder to mediate between several suppliers.	The supplier involves the necessary other parties on his own initiative.	
2. Direct contact: In the period [from 8:00 to 18:00 on weekdays], users can quickly contact a supporter by phone or in person.	In this period, contact is available within __ minutes. (The customer expects 10 minutes.)	
3. For a direct contact, the supporter handles the request on the spot as far as possible.	[On the spot means what can be done within 5 minutes.]	
4. Indirect contact: Requests sent by email, sent by web, or escalated from the direct contact get a reply within a few hours.	[The supplier replies within __ work hours (8:00 to 18:00 on weekdays). (The customer expects 3 hours.)]	
5. The supplier sends a supporter when this is necessary to remedy the problem.		
6. The supplier can perform remote diagnostics to remedy the problem.		
7. The supplier monitors request handling to see that requests are closed and response times met.		
8. The supplier records data for computation of support response time, and identification and prevention of frequent problems.	The supplier keeps a log of all steps in the request handling and the cause of the problem.	
9. The supplier monitors the operation in order to foresee availability problems, and changes the technical configuration so that availability is maintained.		
10. Customer and supplier meet regularly to review response times and discuss prevention of problems.	The parties meet every ____. (The customer expects monthly meetings.)	

L5. Maintenance

This section shows examples of typical maintenance requirements, including defect removal, system updates, and system expansion.

Who decides whether a reported defect is urgent (business critical)? Is it the user who reported it or the supplier? The answer depends on the kind of system and customer we deal with. Usually it is not the user because users tend to consider everything urgent. On the other hand, the supplier prefers to deny that it is urgent.

Requirement 2 suggests some alternatives. The basic one is that the supplier decides, and that his decisions are reviewed regularly (requirement 4). The alternatives are that the local super user or the customer's IT department decides.

When a system is to be expanded, the supplier has a de-facto monopoly and can charge the customer accordingly. Requirement 6 shows a way around it: The size of the change is estimated as the number of *Function Points*, and the supplier has specified a fixed price per Function Point. COSMIC points are a modern version of Function Points. They are much easier to use.

Both methods are a technology-independent way of measuring development tasks. Depending on the supplier's skills, technology and system documentation, he can quote a higher or lower price per point.

Expertise is needed to estimate Function Points, and disagreements are likely. Many countries have a special interest group for Function Points, and requirement 7 specifies that it must be used to resolve conflicts.

L5. Maintenance

Maintenance includes defect removal, system updates and system changes. The specified times must apply for [95% of the cases].

Requirements for defect removal:	Example solutions:	Code:
1. The supplier keeps a log of reported defects as well as change requests.		
2. For all reported defects, the supplier quickly decides whether the defect is business critical, possible to circumvent temporarily, or possible to circumvent permanently (i.e. reject). [Alternative 1:] The local super user decides. [Alternative 2:] The customer's IT department decides.	[In the period from 8:00 to 18:00 on weekdays, the supplier completes the assessment within __ hours. (The customer expects 3 hours.)]	
3. Business-critical defects are removed quickly.	[Business-critical defects are removed within __hours. (The customer expects 24 hours.)]	
4. Customer and supplier meet regularly to check the defect assessments, and to decide what to repair or change, and what it will cost.	[The parties meet every ____. (The customer expects monthly meetings.)]	

Requirements for system improvement:	Example solutions:	Code:
5. The supplier installs new versions and releases of the delivered software within a few weeks.	[Installation takes place within ___ days after release of the new version or release. (The customer expects 30 days.)]	
6. Within a period of 3 years, the supplier must offer changes at a fixed price per [Function Point / COSMIC point.]	The price per [Function Point / COSMIC point] is ____.	
7. Disagreement on the [Function Point / COSMIC point] calculation must be resolved by …		

7. Literature and other templates

Alexander, Ian & Beus-Dukic, Ljerka: Discovering Requirements - How to Specify Products and Services. Wiley, 2009, ISBN 978-0-470-71240-5. Provides good advice and examples of many methods and notations. Contains cases from several domains.

Bon, Jan v., et al. (eds. 2004): IT Service Management - an Introduction based on ITIL. Van Haren Publishing, ISBN 90-77212-28-0. Describes in a comprehensive way the processes associated with operating and supporting a system (240 pages).

Card, Stuart K. et al. (1980): The keystroke-level model for user performance time with interactive systems. *Communications of the ACM,* 23 (7), pp. 396-410. Breaks down the user part of the task into basic elements and measures the time for each type of element.

Constantine, Larry & Lockwood, Lucy A.D. (1999) Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design, Addison-Wesley. Describes a systematic design method for user interfaces, starting with elicitation of essential use cases and ending up with prototypes and usability testing.

COSMIC, Common Software Measurement International Consortium. A modern method for functional size measurement. It is applicable to business, real-time and infrastructure software. The method is entirely 'open'; all method documentation is available in the public domain for free download.
http://www.cosmicon.com/

Lauesen, Soren (2002): Software Requirements - Styles and Techniques. Addison-Wesley, ISBN 0-201-74570-4. A textbook on how to formulate requirements, elicit them, assess solutions and test them. Contains large sections of real-life specifications formulated in different ways. See:
http://www.itu.dk/people/slauesen/SorenReqs.html

Lauesen, Soren (2005): User Interface Design - A Software Engineering Perspective. Addison-Wesley, 0-321-18143-3. Shows how the designer gets from task descriptions and data model to a user interface that meets the usability requirements. Answers the difficult question: How many screens are needed and what should they contain? See:
http://www.itu.dk/people/slauesen/SorenUID.html

Patton, Ron (2006): Software testing. Sams Publishing, Indiana. ISBN 0-672-32798-8. Covers many kinds of test such as white box test, black box test, compatibility test, foreign-language test, and security test.

Robertson, Suzanne & Robertson, James (1999): Mastering the Requirements Process. Addison-Wesley, ISBN 0-201-36046-2. Explains the author's Volere approach by means of a specific example, a system for managing roads in winter time. It mainly covers systems to be developed from scratch. The Robertsons' templates are available on
http://systemsguild.com/GuildSite/Robs/Template.html

Technology Group International: Software Selection Requirements Template (accessed May 2011). A template for comparing business systems (ERP systems) according to around 1250 functional requirements on "product level". You have to register, but then the template is free.

http://www.tgiltd.com/erp-software-selection/erp-requirements-template.html

Wiegers, Karl E. (2003): Software Requirements, 2nd Edition. Microsoft Press, ISBN 0-7356-1879-8. Covers many aspects of requirements from rights and obligations to tools, notations and processes. Illustrated with good and bad requirements, and dialogues from the elicitation process.

Withall, Stephen (2007): Software Requirement Patterns. Microsoft Press, ISBN-0-7356-2398-8. A comprehensive set of things to consider and examples of requirements in many areas. All requirements are on *product level*, i.e. solutions rather than true demands. Usability, for instance is absent.