Guide to Requirements SL-07

Template with Examples v4

Soren Lauesen 2016

Buy the guide as a handy booklet on www.amazon

ISBN: 978-1523320240

Soren Lauesen
Guide to Requirements SL-07 - Template with Examples v4

Version 4, 2016
© Soren Lauesen, 2016

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)

Contents

Background

IT developers and consultants often ask for an exemplary requirements specification as a starting point for their specific project. SL-07 is such a specification. It is a template filled out with a complex example: *requirements for an Electronic Health Record system (EHR)*.

This booklet explains why the requirements are written the way they are, what to be careful about, how the requirements relate to the contract, etc.

You can download the requirements template here:

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

Requirements SL-07 is based on experience with public IT tenders according to the EU rules, in particular when the system is COTS based (Commercial Off-The-Shelf) so that large parts of it may exist already. Later, SL-07 proved advantageous for other kinds of acquisition too, and also 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 Vibeke Søderhamn, Bo Gad Køhlert and Anders Lisdorf for careful reviews of the documents.

Earlier versions of the template have been used with success in more than 100 very different projects, tender processes as well as in-house projects, agile as well as waterfall, for instance: management of home care in a municipality, including route optimization; a pharmaceutical company's innovative document management system; electronic health records; stock management for movie production; claims management for car insurance with GIS as documentation.

Experiences from these 100+ projects helped me write this version of the booklet - version 4. Many things have been improved, for instance integration requirements and supplier selection.

I have experienced that SL-07 works extremely well in practice - once you have learnt how to use it. Although it looks easy, most people get it all wrong the first time, particularly the tasks in Chapter C. With 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, January 2016
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 in detail what the system shall do, 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 becomes the supplier's proposed solution. Initially column 2 is empty or shows a solution example imagined by the customer. 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. Parts in **red** are advice to the customer that isn't intended for the supplier. Delete them.

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 covered by the requirements in one way or another. Often they ask for a specific solution. Write it in column 2 and make sure it doesn't become a requirement in column 1.

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 is costly, but important in an EHR system. In most other projects it is not necessary.

Look at each requirement and ask: What would happen if we got a system that didn't meet this requirement? If it doesn't make a difference, the requirement is superfluous.

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 to carry out a piece of work. Task descriptions resemble "use cases" but don't specify who does what. 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.

> 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 discharge letter? 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. A proposal 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 the 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 customer's needs

In practice we see many systems that meet all requirements, yet are unsuccessful. The user needs are not covered, nor are the business goals.

We can ensure that the user needs are covered by describing the user tasks to be supported by the system, and check that they actually are supported. If we wrote the requirements on product level or design level, we might get a system that did what we asked for, but didn't support the tasks efficiently.

It is harder to cover the business goals. 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 B2 of the template provides a simple way to trace business goals to requirements. Used properly it can help you identify 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 B3 of the template asks for an early proof of concept in order to mitigate these 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 design task descriptions (Chapter C) for these future processes and carry them out as role plays to check that the tasks "work" correctly.
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 record the issues in a list so that the team can progress. *Requests for change* is another name for *issues*.

Review the issues regularly and decide whether to transform them into requirements, into possible solutions, reject them, or keep them in the list. 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. For instance what to do if 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 the example in section A2). 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, for instance some forgotten fields in the database. 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 B3). 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.

When 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: If the requirements say "or better", it is an advantage to exceed expectations.

4. Assessing proposals

In public EU tenders the customer must assess the proposals on a numeric scale and choose the winner with the highest score. 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 work area, carry out the task with the supplier's proposed system. Take notes of 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 B4 and B6 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 B5 and B6 of the template show ways to combine the many scores into a score for each requirement area, 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.

4.1. 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, many tender processes don't allow the supplier to specify alternatives.

On the other hand it is risky to forbid alternatives. The requirement in section A2 shows a real-life example where the customer had required a system availability of 99.5%. The supplier had two operations options (99.0% at 0.5 m$ a year and 99.8% at 2.5 m$ a year), but were not allowed to offer alternatives. So he offered 99.8%, which the customer accepted. The customer could easily have done with 99% and thus lost $2 million a year because he didn't allow alternatives.

If the customer uses a selection approach with a modest number of sub-criteria (like section B5 and B6), 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. For the other alternatives, assess the marginal effect on the final score (including costs).
2. If there is a clear 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 entire proposal.

In the A2 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 must ensure this.

4.2. Options

Above, the supplier defined the alternatives. The customer may also define some; they are called *options*. As an example, the customer may want a data warehouse as part of the delivery and asks for a separate price for it. Should he say yes or no to the data warehouse? Should it have an effect on whom he chooses?

He can make this difficult decision in the same way as for the alternatives: Calculate the marginal effect on the final score and say yes to the option if it has the highest score.

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 and real users.

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 14, 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 your own requirements 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 14). 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**. These parts must be replaced with something else in the final requirements - or deleted. Parts in **red** are warnings or alternatives. Delete them. Other parts can 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 5.4 01-01-2016, 21:59
Last changed by: slauesen

Requirements specification for Electronic Health Record System

(below called the EHR system)

Customer
Midland 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 B2 elaborates the business goals). Briefly explain the customer's present situation and his visions about the future.

The suppliers like some figures about the customer in order to get an idea about the "size" of the project. How many users, how much data, etc. Write a few key figures.

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 an EHR system including a medication system. This is indicated with the double-border box (the delivery) that contains the medication system. Maybe his own EHR system already contains a medication system, or he choses one (maybe the customer's present one) and integrates with it.

He also has to integrate with the existing SKS tables 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.

Midland Hospital has around 5,000 employees, 800 of which are doctors. The hospital has around 50,000 in-patients a year and around 200,000 out-patients.

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 B2).

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

Figure 1: Existing system

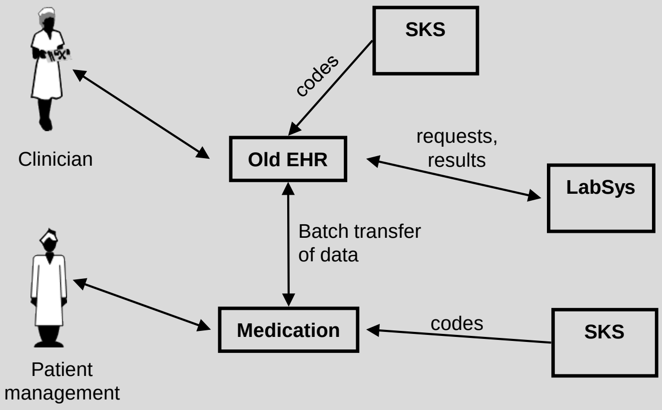
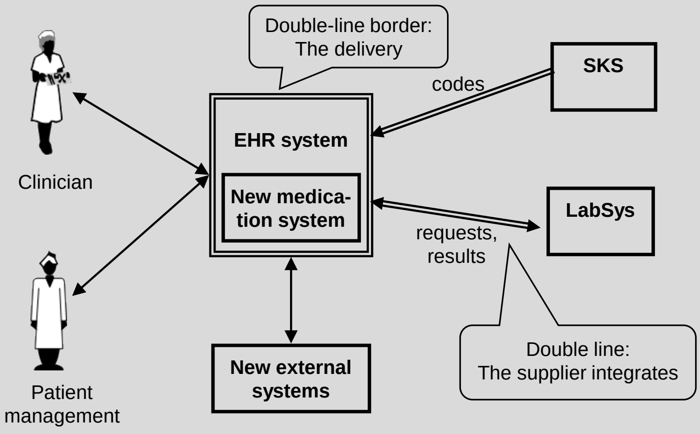


Figure 2: Vision for the new system



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, what are solutions and what are assumptions the supplier can make.

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.

We recommend that the supplier's proposal is in **red**. After several experiments with various indications such as *italic* or a color per author, it was obvious that red for the proposal was the most distinct and legible. It is also far easier to read than traditional tables with a column for each party or - worse - one appendix for the requirements and another for the proposal.

We have seen column 3 (code) being used in many ways, for instance:

1. Requirements priorities.
2. The supplier's indication of whether the proposal is part of a COTS system, an extension, a later delivery, etc.
3. The customer's score for the proposal.
4. Later in the project a reference to a test case that tests the solution to this requirement. In this way you ensure that all requirements are tested somewhere.

You may change the supplier guide to show what the code column is to be used for in this project.

Examples in the supplier guide

The concepts of *alternatives* and *open target* are illustrated with a requirement on system availability. There may be a need for more examples.

In earlier versions of the requirements template, the concepts were illustrated with an imagined example (a system for support of a hotline). The intent was to reuse it in you own project without change. In practice this caused a lot of confusion.

Recommendation: Extend the supplier guide with a few requirements from you own project, and show how the supplier in principle could reply to them.

A2. Supplier guide

This section explains the requirements format.

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 contain the customer's solution example. In the supplier's reply, column 2 is a short description of the proposed solution. It must be in red.
- Column 3 may be the customer's rating of the proposed solution, test references, etc.

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.

The customer's solution examples are only for inspiration. The supplier is welcome to suggest completely different solutions. They become legal requirements when both parties have accepted them. However, in case the accepted solution doesn't meet the demands stated in column 1 in a reasonable manner, column 1 has priority.

Text outside tables

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.

Alternatives

Customers often write requirements that turn out to be very expensive to meet. In such cases, the supplier is welcome to offer alternative solutions: an expensive one that fully meets the customer's requirements and a cheaper one that only partially meets them. The requirement in the table below is an example.

When the proposal has alternatives in several areas, it is important that the customer can assess them independently.

Open target

Chapter L has many "open target" requirements. As an example, the customer may ask for high system availability, but isn't sure what it will cost. So he states what he expects and leaves it to the supplier to suggest something. In the proposal it may look like this:

Availability requirements:	~~Example solution~~ Proposed solution:	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 at least ____%. (The customer expects 99.5% or better). Alternative A: 99.0%, around 0.5 m$/year, see app. . . . Alternative B: 99.8%, around 2.5 m$/year, see app. . . .	

Notice that the customer has written "99.5 *or better*". It means that the supplier earns additional points for alternative B. In case the supplier had omitted "*or better*", alternative B wouldn't give more points than 99.5% would, only additional expenses.

The template format

The template is an MS-Word document. It uses *Heading 1, Heading 2* and sometimes *Heading 3*, plus a special heading style, *Heading no number.* They automatically generate the table of contents. In order to improve the overview, some headings have a forced page break. It may be changed through

Home → Paragraph → Line and Page Breaks → Page break before

Tables use the table style *Requirement Table.* It has borders of ½ 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

This chapter doesn't contain real requirements, but provides connections between requirements, the customer's business goals and the acquisition process.

B1. Flows

When you observe users, you often see small pieces of work (tasks) that are part of a larger flow that produces the results we care about.

The EHR example has only one flow: Treatment of a patient from admission to cure. On the way you examine the patient, make diagnoses (what are the diseases), plan and perform treatments, check results and discharge the patient. The entire process can take days or months.

A flow is also called a process, a business process, a life cycle or a high-level task or a high-level use case.

In the health sector there are other flows than patient treatments, for instance the life cycle of a treatment from it once upon a time was recommended by a commission until it years later is abandoned by another commission. The EHR system is not supposed to support this flow, but it might deliver data to it.

Flow as a table: In section B1 you describe the flows to be supported. We recommend that it is done as a table, but it could also be graphical. In the example a treatment flow consists of 12 steps, but it isn't sure that all of them are done for a specific patient. Some steps may be repeated several times, for instance check-ups.

Column 2 of the table shows the tasks and subtasks that perform the step. As an example, two different tasks can perform step 1: Admission before arrival (C1, e.g. through the GP) and acute admission (C2, e.g. a traffic accident). As another example, steps 3 to 4 and 6 to 9 are performed by one task, the clinical session (C10).

As you see, there is a many-to-many relation between the steps of the flow and the physical, observable tasks. It is not a hierarchy.

When you describe a flow, you often find new demands for IT support. In this case we detected a need for arranging check-ups (step 8) and for coordination with home care (step 10). These defects are shown as red question marks in the table.

Flow as graph: A widely used graphical notation is BPMN (Business Process Modeling Notation). It shows each step as a box and connects the boxes with arrows to indicate the sequence, and with diamonds to show choices to be made about the sequence. It can give a nice overview - unless you go into too much detail and try to specify also what to do in exceptional cases.

In practice we see a lot of effort being spent on flow diagrams and they occupy a lot of pages. Often it is hard to see whether the flow is about the logical steps or the physical tasks, and even harder to check them against each other, as we have done in column 2 of the table.

B. High-level demands

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

B1. Flows

The system shall only support one kind of flow: treatment of a patient. The table below is the general, logical flow of a treatment. Many of the steps can be omitted (e.g. step 2 and 8) or repeated several times during the treatment (e.g. step 3 to 9).

The logical flow is carried out in physical tasks, where an employee for a short period of time works with the patient without essential interruptions. Column 2 shows the related tasks and subtasks for each step in the flow. Chapter C shows the details.

Steps in patient treatment	**Tasks and subtasks**
1. Admit the patient either through GP (General Practitioner), the patient in person or acutely (e.g. traffic accident with unconscious patient).	C1, C2
2. Call the patient to make an appointment.	C1-4
3. The patient arrives to the ward. Examine the patient to make a diagnosis, including making tests on the spot or through a lab.	C10-1, 2, 3 C12
4. Plan the treatment, including ordering medicine, booking time, order implants, etc.	C10-6, C11, C13
5. Maybe transfer the patient to another ward, for instance in case of several diagnoses.	C3
6. Treat the patient.	C10-3, C14
7. Evaluate the result. Maybe perform further tests and treatments.	C10
8. Make appointments for check-ups.	C10-6 **?**
9. The patient arrives for check-up. Perform various tests and maybe supplementary treatments.	C10
10. Arrange home care.	**?**
11. Discharge the patient and inform relevant parties, e.g. own GP or social services. The patient may also have died.	C6, C7
12. Settle accounts	C8

In the general flow above, we haven't mentioned time monitoring at the various steps. It is described in tasks and subtasks.

The flow description is a cross check between the logical flow and the tasks. In the case above it revealed some flaws in the tasks, marked by question marks above.

B2. 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 the vision 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. The customer may 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. Section B4 explains 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 that you easily could make new treatment plans and new user screens. When they had to fill in the goal table, they realized the demand and came up with the requirements in E4.

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 insist on a goal that isn't 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), but this is hard to relate to a business value. It could also be measured as the number of tasks performed per person per day, or as the time spent at the computer per patient. These are hard data and they relate well to a business value. Goal 4 could be measured as operational costs before and after system deployment.

It is important to have these figures in order to select the most advantageous proposal as described in section B5.

B2. 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 several 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 problem 2p (assess the state of the patient) and 6q (errors at hand-over). 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.	Requirements in sections E4 and F10 (system expansion and integration with new systems).	
4. Lower operational costs.	Acquire a new, hopefully cheaper, system.	All the requirements and the selection criteria in B5.	
5. Meet the new EU rules on ...	…	…	1-1- 2017

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 B6 gives an example of how to avoid it.

B3. 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 B3-1 to B3-5 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. He can include the cost of the proof in his quotation, and will thus be paid when he delivers as promised.

B4. Minimum requirements and selection criteria

How do we select the best proposal? Traditionally, selection criteria are part of the tender material, but not of the requirements. The SL-07 template includes them in order to show how they can be expressed in a way that links to business goals and requirements. Move them to other parts of the tender material as needed.

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, in government acquisitions, this is usually not allowed. The customer might fiddle with the criteria so that his favorite supplier becomes the winner. As an example, the EU regulations state that you must define two sets of criteria and tell the suppliers about them in advance:

A. Minimum requirements. Proposals that don't meet all of these are discarded.
B. Selection criteria. Each criterion has a predefined weight. Each proposal gets a score for each selection criterion. The proposal with the highest weighted sum of the scores gets the contract.

This is hard in practice and may force a customer to select the "wrong" supplier.

B3. 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. Deficiencies here can hardly be repaired late in the project. 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.

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. (See also area 5 below.)	
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.	

Problem A: There may be more than thousand requirements. Which of them are mandatory (minimum)? If all of them are mandatory, you may discard good suppliers that fail on some unimportant requirement. If only some of them are mandatory, the customer may be forced to select a supplier that fails on the rest.

The template deals with the problem by defining minima for requirements **areas** rather than individual requirements (see example below). There may be around 40 areas, and it makes sense to check whether each area is adequately supported. The area may be adequately supported although some detailed requirements are not supported. Typically, some suppliers fail on certain requirements in the area, others on other requirements. They can still be considered as long as they support the area as such adequately.

Problem B: How do we define scores and weights in practice? Some analysts use weights that add up to 100%, but how can these be justified? Usually the figures have no relation to the business value, and as a result the customer doesn't select the proposal with the highest business value.

SL-07 shows two solutions: One where the selection criterion is the net business value (B5), and one where it is the number of score points per dollar (B6).

Minimum requirements and minimum scores

In the request for proposal, the customer has divided the requirements into areas and specified a minimum score for each area. In this way we cover all the requirements. There are around 20 areas in the EHR example.

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 EHR example:

Area	Minimum scores	
C1-C4	Admit patient. Support is not really needed for this task. The customer can just keep his existing admission system.	-2
C10	Perform clinical session. To avoid selecting a supplier who scores high elsewhere, but handles clinical sessions badly, we demand that the system supports clinical sessions at least as well as the present system.	0
C11-C…	Medication (considered one area). This too must be supported at least as well as today.	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 - it is one of the business goals.	1
H1	Login and access rights for users. This must be at least as good as today.	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.	0
J2	User training. This must be at least as good as today. It indirectly gets a financial score too 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.	0
…	…	…

When the suppliers have submitted their proposals, the customer must assess each proposal. Chapter 4 explains how to give a score for each task, each integration, etc. and take notes about it. Give a final score for each **requirement area** based on the scores and notes. A single bad score for one of the tasks may give a bad score for the entire area. Make a note of why the area is not supported adequately.

Minimum requirements: All requirement areas must meet the minimum score.

The notes are useful for internal customer discussions. They are also useful in case a supplier finds the assessment unfair and goes to court. The notes can prove that the customer actually made a fair assessment.

The principle of giving a final score for each requirement area rather than for each requirement is important. It is virtually impossible to assess a requirement in isolation. Requirements interact and together they give a more or less good support of each requirement area. Good support of requirement A plus bad support of requirement B may have the same business value as bad support of requirement A plus good support of B. So they should get the same score.

Don't let the supplier write the scores. Amazingly many customers have a table of all the requirements and ask the suppliers to fill in to which degree they meet each requirement, e.g. met/not met. *We don't have the time*, says the customer, *let the supplier do it*. Imagine what suppliers do in this case? They let their sales department fill in the scores. Not surprisingly, all suppliers get top scores for everything. As a result the customer has to choose based on the cost only, and often ends up choosing a bad system.

The consequence may be that the customer saves some work hours now, but wastes thousands of hours later because his staff has to work with a bad system.

In the EHR example, some of the areas are assessed twice: When selecting the winner and during the early proof of concept. This is okay. In this way the customer gradually reduces the risk that the supplier cannot deliver as promised. If the solution fails during the early proof, it is a good reason for terminating the contract (see section 3.2).

B4. Minimum requirements

Sections B4 to B6 are 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.

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

Minimum score: For each requirement area, the customer has stated the minimum scores below. A system that that doesn't meet the minimum scores, will be useless in practice.

Minimum requirements: The system must meet the minimum scores below on all requirements areas.

Notice that a minimum score may be -2 or -1. This means that a proposal may be acceptable even if it is worse than the present system in this area. As an example, area C1-C7 has a minimum score of -2 because the customer can use his existing admission system. The table shows a fictitious example where proposal A scores -1 (worse than today) for area H2-H5, but this is acceptable because the minimum score is -1.

Requirement area	**Minimum score**	**Score**		
		A	**B**	**C**
C1-C7. Admit and discharge patients (considered one area).	-2	1		
…				
C10. Perform clinical session.	0	1		
C11-C… Medication (considered one area).	0	2		
…				
D. Data. Assessed through the task support.	N/A	N/A		
…				
F10. Integration with new systems.	1	1		
H1. Login and access rights for users.	0	0		
H2-H5. Other security (one area).	-1	-1		
I. Usability and design.	0	1		
J2. User training.	0	0		
J4. Data conversion.	0	1		
L1. Response times.	0	0		
…				

B5. Benefit in dollar

The minimum requirements discarded the proposals that were unacceptable. Among the rest we have to select the winner.

Method B5 computes the net benefit in dollars for each proposal. The customer selects the proposal with the highest net benefit.

First the customer computes the potential value of each business goal. In the example, the customer has computed the potential value for a period of 5 years. As an example, efficient support of the tasks might save each employee an hour a day. This estimate is based on observations of clinicians at work. As an example, clinicians today have to log into several systems for each patient and take paper notes to get an overview. It might be avoided with proper system integration and overview screens. For 4000 clinicians, it means saving 200 million $ in 5 years.

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, we compute the five-year value 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.

Method B5 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.

B5. Selection criteria: Highest net benefit

Use either section B5 or B6 as selection criteria.

The total benefit of the proposal is based on a financial value for each business goal. The table shows an example with fictitious figures for proposal A.

Potential: The customer's estimate of the potential benefit for a 5-year period. Measured in million $.

Fraction: For each proposal the customer estimates the fraction of the potential benefit that this proposal can reach if the supplier delivers 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. Proposal A seems to save only half an hour and gets the fraction 0.5.

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

5-year value: Computed as Potential * Fraction * (1-Risk)

Business goal	**5-year poten-tial**	**Fraction**			**Risk**			**5-year value**		
		A	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
1. Efficient support of clinical tasks	200	0.5			30%			70		
2. Reduce medication errors	50	1.0			10%			45		
3. Continuous improvement	50	1.0			40%			30		
4. Lower operational costs (included below)										
Total benefit for 5 years (million $)	**300**							**145**		

The customer estimates the net benefit for each proposal. The total benefit for a period of 5 years is computed above. 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 differ between proposals.

The customer selects the proposal with the highest net benefit for 5 years.

Benefit for 5 years, million $	**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		
Total costs for 5 years	**55.6**		
Net benefit for 5 years	**89.4**		

B6. Benefit in score points

Method B6 also discards the proposals that were unacceptable according to the minimum requirements. However, it doesn't calculate the benefit in dollars, but as a weighted sum of score points.

Total weighted score points: We start with a copy of the table for minimum requirements and replace the minimum scores with a weight for each area. We keep the actual scores for each proposal. We add columns where we calculate the weighted score for each requirement area and each proposal. The total weighted score for a proposal is an indication of the value of the proposal.

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.

Instead we could find weights that reflect the size of the area, for instance the number of staff affected, the effect on quality, or the effect on cost. In the example we started with the 5-year potential value computed as in B5. We split the potential value into the requirements areas. As an example, *continuous improvement* originated mainly from F10, *integration with new external systems*. Finally we disguised the values as a weight by dividing by a factor that made the weights add up to 100. This also matches the tradition of using weights that are percentages.

Some areas are not business goals. Yet they have got a small weight that reflects some subjective value anyway. Notice that many areas have weight zero. Better support of them has little impact - as long as the minimum score is met.

In the example, C10 has a very high weight because it accounts for almost half of the business value. This makes the result very sensitive to the score being one or two. For this reason the example gives scores with one decimal for C10. The decimals can be computed based on scores for the individual tasks or requirements.

Total cost: The cost in method B5 is computed exactly as for method B5.

Bottom line: In B5 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 B6. 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 weighted score points per million dollars.

Method B6 will reject the proposals that don't meet the minimum requirements, and among the rest select the one with the highest weighted score per million dollars.

Comparison: The main advantage of B6 is that we don't have to reveal the business value to the suppliers or to the government body that funds us. B6 also allows us to put weights on quality aspects that cannot be estimated in dollars. Finally, the whole procedure is somewhat simpler because we can reuse the scores from the minimum requirements.

Although we don't select the supplier based on the business value in dollars, it is still useful to estimate the business value and transform it to weights.

B6. Selection criteria: Most score points per dollar

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 are not included below, but it could be done.

Scores: The scores are those the customer assessed for the minimum criteria in B4. Since one of the areas has a very high weight, the decision is very sensitive to this area getting score 1 or 2. For this reason we give it a score with one decimal here.

Weight: Each requirement area has a weight that reflects the impact of the area. For instance the number of staff affected, the impact on the customer's service quality, or the contribution to the business value. The weights add up to 100.

Requirement area	**Weight**	**Score**			**Weighted score**		
		A	**B**	**C**	**A**	**B**	**C**
C1-C4. Admit patient (considered one area).	5	1			5		
…							
C10. Perform clinical session.	50	1.5			75		
C11-C… Medication (considered one area).	15	2			30		
…							
D. Data. Assessed through the task support.	N/A	N/A					
…							
F10. Integration with new external systems.	15	1			15		
H1. Login and access rights for users.	0	0					
H2-H5. Other security (one area).	0	-1					
I. Usability and design.	10	1			10		
J2. User training (included in the costs below).	0	0					
J4. Data conversion.	0	1					
L1. Response times.	5	0					
…							
Total weight and total weighted score points	100				135		

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.

The customer selects the proposal with most score points per million dollars.

Score per million $	**Total A**	**Total B**	**Total C**
Total weighted score points	135		
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 points per million $	**2.4**		

Variations

There are many variations on the selection themes above.

For the minimum criteria, 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 worse than today in a few areas. However, it shouldn't be worse in too many areas. We might deal with this by means of an additional criterion: It may not be weaker than **today** in more than 3 of the 30 areas. (It still has to be at least as good as the stated minima for all areas.)

We might add maximum criteria on the cost, e.g. *our budget doesn't allow us to invest more than 30 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.

In B5 we could change the 5-year period to for instance 10 years. This will make the selection less sensitive to the development time and the initial costs.

In B5 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.

For B6 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.

We can vary the score scale, for instance from (-2, -1, 0, 1, 2) to (1, 2, 3, 4, 5). This will make the B6-selection more sensitive to cost differences and less to quality differences as shown by the example below. The B5-selection doesn't have this weakness. 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.

Effect of changing the scale: Assume that proposal A has 100 weighted score points when we use a scale from -2 to 2. If we change the scale to 1 to 5, we must add 3 to each score. Since the weights add up to 100, the total weighted score points increase by 300, as shown in the table below.

Proposal B has 0 weighted score points on scale -2 to 2 and gets 300 when we use the scale from 1 to 5. However, proposal B is also cheaper. As the table shows, A will be the winner with scale -2 to 2. B will be the winner with scale 1 to 5.

Assuming that the score points reflect business values, then scale -2 to 2 better reflects the total business value. Proposal A adds value, B doesn't. On scale 1 to 5 it looks as if proposal B adds value. It doesn't.

Scale effect	scale -2, -1, 0, 1, 2		scale 1, 2, 3, 4, 5	
	A	B	A	B
Total weighted score points	100	0	400	300
Cost, millions	200	100	200	100
Scores per million	0.5	0	2.0	3.0

C. Tasks to support

This chapter describes the user tasks to be supported. The requirement is that all tasks must be supported to some degree.

A user task is something user and computer do together from start to end without essential interruptions. A good starting 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 for the client right now - the user deserves a "coffee break" (*task closure*).

The first task in the template is C1. It starts when the secretary receives a message about a patient. It ends when the patient has been admitted and got a meeting time - or put on the waiting list - or the call has been parked because some information is missing.

The table lists the sub-tasks 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.

Subtask 1a is a variant, i.e. another way to do subtask 1. Either 1 or 1a is done.

In a task you can specify that something is a **problem** to be eliminated. You don't have to specify how. In the example it is a problem that some of the electronic messages don't observe the MedCom format.

Strictly speaking, we must distinguish between **task description** and **task execution**. The secretary executes C1 many times a day. The first time it is about patient A, the next time it is about patient B who lacks information and is parked. At the end of the day the secretary receives another message about patient B, this time with the missing information. Now the secretary can do more for B than the first time. Each time it is a new *task execution* but it follows the same *task description*. Programmers would say that C1 is a *class* and execution of C1 is an *instance*.

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 we often have to repeat the specification. So it is convenient to bundle tasks according to user kind and environment. 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.

User profiles are a short version of **personas**. Tasks are related to **use cases** and **user stories**, but are less solution oriented. See more below.

C. Tasks to support

The system must support all user tasks in this chapter, including all subtasks and variants, and mitigate the problems. Column 1 of the tables describe what user and system will do together. Who does what depends on the chosen solution.

A task is carried out from start to end without essential interruptions. If necessary, the case must be parked and resumed later. Although subtasks are numbered, 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.

Some subtasks may be performed in alternative ways. It is shown with a, b, etc. Letters p, q, etc. indicate something that today is a problem with this subtask.

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 easily with medical staff.
User profile:	**Clerical staff** …

C1. Admit patient before arrival

This task creates an admission or continues a parked admission. Most admissions can be handled as one piece of work. 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 E1-1)

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 formats.	
3p.	**Problem**: Some electronic messages don't follow the MedCom format.	The system allows manual 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 or a confidential email.		
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.		

C1. Task rules (Admit patient before arrival)

The task description consists of these parts:

ID: Tasks are numbered C1, C2, etc. To avoid too much renumbering during requirements elicitation, we group tasks according to work area and start each area with a round figure. In the template, C10 is the first task in the next work area, *patient treatment*.

Name: A task must have a name in imperative, e.g. *Admit patient.* Names like *The user admits the patient* or *patient admission* don't start with a verb in imperative. Imperative hides who does what - user or computer. During elicitation we don't yet know who will do what.

Introduction: Help the reader understand what the task is about.

Start: A task should be something that is carried out from start (trigger) to end (coffee break) 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. In the example it is that a secretary receives a message about a patient. Avoid out-of-the-blue triggers such as *the user wants to record an admission*. This reveals that we haven't understood when this happens and whether the system could support it better, for instance through automatic receipt of MedCom messages.

End: A task ends when the user deserves a "coffee break", either because the user has done what is needed or because nothing more can be done right now. Task C1 may be parked because some data are missing (subtask 4b). Although the task isn't completed in a logical sense yet, it is completed physically 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.

Why do we define tasks this way? Because it is the observable period of time where the system must support the user - without essential interruptions. We must check that the system provides efficient support for the entire period.

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. These figures are outside the table, meaning that they are not requirements, but assumptions the supplier can make. The corresponding quality requirements are in other sections: 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. *Difficult* is outside the table and thus not a requirement. Early in the requirement process you may write *difficult*, but try to move it into other sections 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.

C1. Admit patient before arrival

This task creates an admission or continues a parked admission. Most admissions can be handled as one piece of work. 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 E1-1)

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 formats.	
3p.	**Problem**: Some electronic messages don't follow the MedCom format.	The system allows manual 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 or a confidential email.		
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.		

Users: The users who carry out the task, the environment, etc. Omit this information if there is only one user profile in the work area.

Subtasks and variants: The requirements are in the table. Column 1 is a list of subtasks, variants and problems. This is what the system must support. Subtasks are numbered in a logical sequence, but this is for reference only. Subtasks are basically optional (need not be carried out), some of them are repeatable and they may be carried out in many sequences. The user decides as far as possible what to do.

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, write a requirement note below the table.

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, paying an amount. 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.

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. Solutions are not in imperative. They should explicitly state who does what, e.g. *The system shows* or *The user selects*.

What are the requirements? Right after heading C in the template, you see that the requirements are to support all user tasks, including all subtasks and variants, and mitigate the problems. This means that column 1 of the tables are the requirements (the customer's demand). Column 2 may contain a solution example, but the solution is not a requirement. Things outside the tables are assumptions the supplier can make or help to the reader. Requirements or solution examples that are too long to fit in the table, may be written outside the table, but must have the heading *requirement note* or *solution note*.

C2. Similar tasks (Admit immediately)

Task C2 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 C1 and C2.We need to check that they are supported well in all 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. A complex task (Perform clinical session)

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 without essential interruptions. It is important that the computer supports this mix well.

Many patients have several diagnoses (diseases) and during the clinical session the clinicians may try to deal with all of them. They may for instance follow up on a treatment of one disease and plan treatment of another one.

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.

C2. Admit immediately

This task creates an admission for a patient who arrives in an emergency without prior notice …

Work area 2: Patient treatment

This work area comprises …

C10. Perform clinical session

A clinical session may comprise diagnosis, planning, 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: Disaster with many injured. (Better describe it as a separate task. See the guide booklet.)
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 related indications. See notes. See results of services ordered earlier and compare them with expectations. The data to overview comprises D1 …	The system shows an overview of everything on one screen, e.g. with a Gantt-like time dimension. The user can drill down to details from the overview.	
2p. **Problem**: Today clinicians have to log in an out of several systems to see all relevant data.		
3. Provide services that can be given on the spot, e.g. 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.		
5p. **Problem**: Cumbersome to see standard recommendations.	The system can show recommenda-tions and checklists based on a selected diagnosis.	
6. Plan and order new services. Check against available time for all parties - including the patient. (See the long subtasks C11, C12 … for prescription, booking …).	For bookings, the system shows available dates and times for all parties.	
6p. **Problem**: Parts of the request are forgotten.	The system can book standard packages of services.	
6q. **Problem**: Errors when data are 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. A long subtask (Prescribe medicine)

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 divided into 10 groups. 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*, C12, *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, C10. 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.

C20. Another environment (Perform clinical session, mobile)

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 clinician moves 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 introduction to C10 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 of C10? And how will the customer assess the solution? We suggest that you repeat the task for each environment:

C10: Perform clinical session, stationary.
C20: 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 all contexts.

Tasks, user stories and use cases

User stories

Like tasks, user stories try to describe what the user will use the system for. We see User Stories being used more and more in requirements. They come in many versions. Here are three typical examples:

A. As the patient's doctor I want to see an overview of the patient's diagnoses.

B. As the patient's doctor I want to see an overview of the patient's diagnoses. (Followed by a screen with an outline of the overview).

C. In order to see the patient's diagnoses, I right click the patient's name and chose *See diagnoses*.

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 and diagnoses.	
1p. **Problem**: Cumbersome to see standard recommendations	The system can show recommenda-tions and checklists based on diagnosis and drug type.	
...		
6. Calculate dose. Check that it is reasonable. Check for interaction with other drugs.	The system offers a calculation based on body weight retrieved from the health record.	
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.	
...		

. . .

C20. Perform clinical session, mobile

Clinical sessions may be performed when medical staff is moving around from patient to patient, e.g. with a PDA 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:

All three examples are unsuitable as requirements. You cannot see what the overview is to be used for. Is it to find a treatment, to explain a new symptom, or to write a discharge letter? You may get support for these user stories, yet fail to get adequate support for the full task from trigger to coffee break. In the worst case the user may have to write the diagnoses on paper in order to execute the next steps of the task.

Examples B and C are on a wrong level (design level) and prescribe a specific solution. It is not suitable if you want to get an almost finished system.

Use cases

Use cases also try to describe what the user will use the system for, but are also very solution oriented. They have a short time span and rarely stretch from the true trigger to the deserved coffee break.

In use cases you don't mention problems. Some use case authors become upset if you do it: *You haven't done your work properly*. (Yes, but isn't it the supplier who is supposed to solve the problems?)

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 these 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 a scaring real-life example from the hospital world:

A harmful specification from a real EHR project - 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 diagno-sis.	
. . .	

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 prescribes a specific user dialog. Notice the out-of-the-blue trigger: *The user wants to . . .* It is an indication that we don't know when this is done.

As for user stories, we cannot see the purpose of this use case.

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 such as *Create diagnosis* (4 pages) and *Change diagnosis* (3 pages). They referred to almost the same data. It was hard to use the data names consistently in all the use cases. The original specification had also a log-in use case and a select patient use case.

With SL-07 you describe data separately in Chapter D. From the subtasks you may briefly refer to the data that are relevant in this subtask. The template shows examples in C1-3 and C10-3.

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.

Lauesen & Kuhail, 2010, have a detailed comparison of use cases and tasks. It is based on requirements specifications from 15 professional teams in 5 different countries.

D. Data to record

This chapter describes the data to be stored in the system. Data may be described in many ways. The template shows five ways:

1. a short explanation of each of the data classes (tables)
2. a data model (also called E/R diagram or Entity/Relationship model)
3. a data dictionary with details of each field, data volume, etc.
4. the contents of some existing tables
5. the contents of existing screens

Unfortunately there is no ideal way to describe data. Some are easy to understand for stakeholders, but hard to make precise and consistent. Others are opposite.

Chapter D of the template starts with a short explanation of the data classes followed by a data model. Next there is a detailed data description (a data dictionary) and finally examples of using tables and screens as requirements.

Data model (E/R)

Data models are great to give overview and consistency. Domain experts can often understand them, but ordinary users find them hard to understand.

Figure 3 in the example is a data model or E/R diagram. Each box is a class of data. Imagine that there is a pile of file cards behind the box. As an example, behind the Person box there is a file card for each person the system deals with. The box symbolizes a card for one single person. For this reason the name on the box should be singular, i.e. *Person* rather than *Persons*.

Next to the box we list the fields on the card. A card for a person contains the person ID, the name and other simple fields. There should not be repeating fields on the card, such as a list of the person's hospital admissions (in database terms: *first normal form*). Data about an admission must be on one card in the Admission box and should not be replicated on the Person card. On a user screen we can show a person plus all his admissions, but not on a file card in the data model.

There are relationships between the boxes, shown as crow's feet. A crow's foot shows that a card relates to one or more cards in another pile. As an example, a person's file card is related to several admission cards (strictly speaking to zero or more admission cards). Reading the crow's foot the other way, one admission card is connected to only one person card.

A dotted box shows that the data in that pile are shared or partly shared with aan external system.

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

The diagram lists the fields (attributes) outside each box to save space and improve overview. In many cases we show only some of the fields and indicate with ... that there are more. Notice that we don't show the tables' *foreign keys*. It is database technology and confuses the users. The crow's feet show what is needed.

A UML class model is very similar to an E/R model, but fields are shown inside the boxes, so boxes become very large. Connectors are lines with cardinality shown as

0:1, 1:*, etc. When a line cannot be straight, it is not a smooth curve, but a broken line. These seemingly small differences make a huge difference when you try to get an overview of a large diagram. Our brain can much easier perceive 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 in Chapter C. In many cases data has to be exchanged with external systems as specified in Chapter F.

Figure 3 is a data model (an Entity/Relationship diagram, E/R) that gives an overview of the data. Each box is a class of data. Imagine a pile of file cards behind the box. The box symbolizes one of the cards. As an example, D5 is a pile that holds a card for each person the system deals with. Next to the box is a list of the fields on the card.

There are relationships between the boxes, shown as crow's feet. A crow's foot shows that a card relates to one or many cards in another pile. As an example, a person can have many admissions, but an admission relates to only one person. Data need not be structured this way in the system, but it must be handled in some way.

The dotted boxes show data that are (partly) shared with external systems.

D1. Diagnosis: Each record contains data about one of the patient's diseases. It corresponds to the National Health Classification (SKS), but there is also a need for recording diseases that are not in SKS or cannot be classified until later.

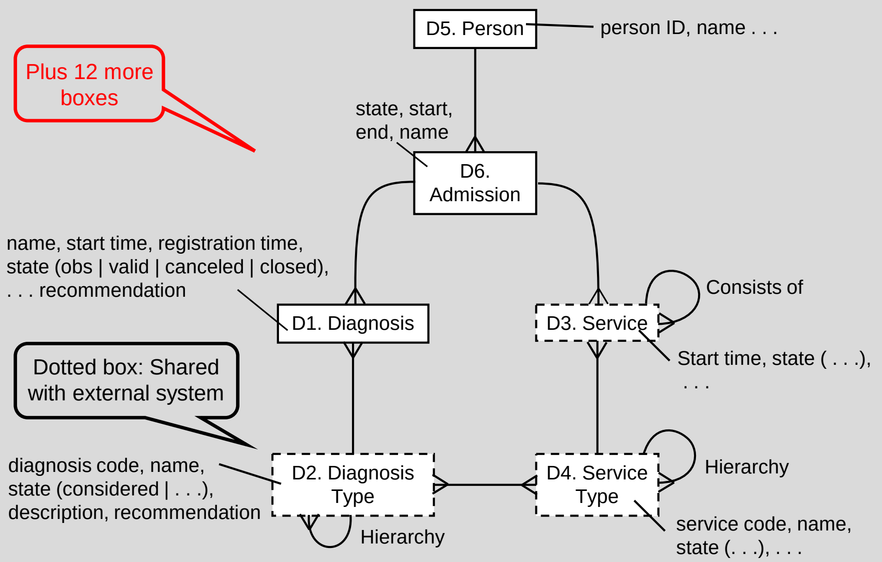
D2. Diagnosis type: Each record contains data about a type of diagnosis - independent of the patient: the diagnosis name and SKS code (where possible), recommendation, standard treatment packages (through the relationship to the catalogue of service types) … The clinicians will choose diagnoses from this catalogue of diagnosis types.

…

D5. Person: Each record holds data about a person: name, address … A person may be a clinician, a patient, or a relative.

D6. Admission: Each record holds data about an admission: start time, related person …

Figure 3. Data model for the system


D5. Person
person ID, name . . .
Plus 12 more boxes
state, start, end, name
D6. Admission
name, start time, registration time, state (obs | valid | canceled | closed), . . . recommendation
D1. Diagnosis
D3. Service
Consists of
Start time, state (. . .), . . .
Dotted box: Shared with external system
diagnosis code, name, state (considered | . . .), description, recommendation
D2. Diagnosis Type
Hierarchy
D4. Service Type
Hierarchy
service code, name, state (. . .), . . .

D0. Common fields

In many systems we need to keep track of the data history, i.e. who created or changed data and when. It is fields that all tables must have. Old versions of the "file cards" are kept. Technically it can be done in various ways, but they are solutions the customer doesn't have to care about. The important part is the requirements in D0.

D1. Data dictionary (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 a file card may represent. Show typical as well as unusual examples. For the diagnosis class, a file card represents a diagnosis for a specific patient. A diagnosis may for instance be "cholera" or "coughing".

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. In the example, the diagnosis name is usually retrieved from a diagnosis type-card, but it may also be entered by the clinician.

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 in the table and thus a requirement. The system must be able to store this amount of data. 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. Attributes are numbered sequentially. Problems associated with an attribute are numbered p, q, etc. The list has three columns, similar to tasks. Notice that we describe the crow's feet (relations) as a kind of field, e.g. D1-2 and 3.

The example is written without details of the data format (e.g. whether it is text or numbers). In some cases details such as date format and text lengths are useful, for instance in the solution column as shown for D1-4. If a specific format is necessary, it must be 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 in the same way as for other chapters of the requirements.

D0. Common fields

Each data class records history, i.e. each change creates a new version of the "file card" and preserves the old one. This is recorded in these fields.

Fields and relationships:	Example solutions:	Code:
1. Change Time: The date and time when the "file card" was created or changed.		
2. Source: The person who created or changed the "file card".		
3. History: Relation to earlier versions of the "file card" (not shown in the diagram).		

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.		

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 type is used for billing and government reporting.		
2p. Problem: Very hard to select the right SKS code from the 20,000 possible ones.	See solution notes below.	
3. Admission: 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 Change 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.	
...		
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 SKS 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. A type class (Diagnosis type)

The diagnosis table D1 holds the actual diagnoses for the patients. In contrast, D2 is an example of a type class. The file cards behind the D2-box make up a 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. The EHR system gets the diagnosis types from the web site of the National Health Organization (SKS), see F1.

Notice how D2-5 mentions an example in column 1 (Cholera DA00). This is a good way to explain what the field may contain. We often see people write such an example in column 2. This is wrong - column 2 is for example *solutions* - DA00 is not a solution.

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.

In the EHR example, there is also a Service Type class (D4 in the model). It corresponds to a catalogue of all possible services, e.g. "Blood pressure measurement" and "Hearth bypass surgery". In some cases there may be several levels of type classes. As an example, doctors don't just prescribe Aspirin. They prescribe the service type "Aspirin, 12 tablet package". This service type belongs to a Drug medicament type that is "Aspirin, 500 mg tablets". The Drug medicament type belongs to a Drug preparation type that is "Aspirin, tablets", which corresponds to an Active ingredient type that is "Acetylsalicylic acid".

The Drug medicament type, the Drug preparation type and the Active ingredient type are separate data classes (separate boxes) not shown in the template diagram.

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 SKS 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. SKS has presently around 20,000 types.		

Fields and relationships:	Example solutions:	Code:
2. Diagnosis code: SKS code 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, e.g. "Cholera, DA00" rather than "Cholera without specification, DA009".	Example: "Cholera, DA00". WRONG - not a **solution** example.	
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. Using existing tables and screens (Service)

In this section we show some other ways of specifying data: Existing tables and existing screens.

There are many kinds of service in an EHR system. It is hard for the customer to specify all of them. In the first part of D3, the customer has specified the common fields and relations that all services should have.

D3-4 is the state of the service. When the clinician requests a service, it starts in state *ordered*, then becomes *confirmed* by the service provider, then *started* and *completed*, and it should end up as *assessed* by a clinician. Keeping track of the state and when it changes, is important for taking action when things don't proceed as expected. It also allows the system to issue warnings when something is forgotten. The rules for changing state and issue warnings can be complex. You may write the rules as requirement notes below the table or as business rules in Chapter E.

Section D3.1 specifies the services that are clinical measurements. In principle, the special fields for a clinical measurement should be written in table D3.1, but D3.1-2 just refers to a screen taken from the existing database system. The screen lists the existing fields. This gives the supplier an idea about what is needed, but the details may have to be sorted out during development.

Section D3.2 specifies the services that are surgery. They are specified in the same way as the measurements.

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 also gives the supplier an idea about what is needed, but again the details may have to be sorted out during development.

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 external systems to be integrated.

Fields and relationships common for all services:	Sample solutions:	Code:
1. Service code: Relation to Service Type.		
2. Admission: Relation to the Admission, which in turn refers to the patient (Person).	The system records it automatically based on the currently selected patient.	
3. Start time: The date and time 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 services.		

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.		

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 these common service fields: admissionID and state.		

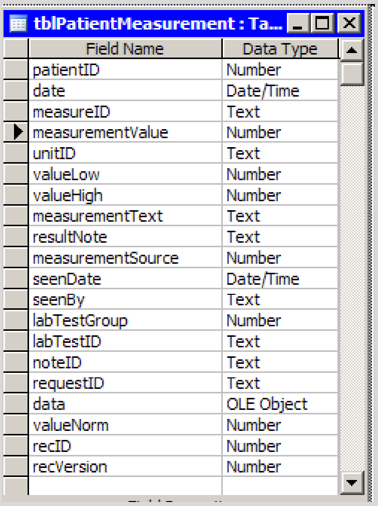
D3.2. Patient surgery

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.

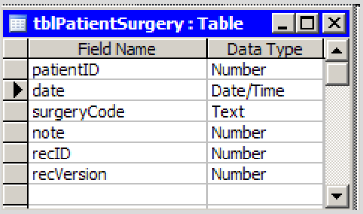
Data volume:	Example solutions:	Code:
1. Around 100 surgeries are recorded a day.		

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 service fields: admissionID and state.		

Figure 4. Present service



Field Name	Data Type
patientID	Number
date	Date/Time
measureID	Text
measurementValue	Number
unitID	Text
valueLow	Number
valueHigh	Number
measurementText	Text
resultNote	Text
measurementSource	Number
seenDate	Date/Time
seenBy	Text
labTestGroup	Number
labTestID	Text
noteID	Text
requestID	Text
data	OLE Object
valueNorm	Number
recID	Number
recVersion	Number



Field Name	Data Type
patientID	Number
date	Date/Time
surgeryCode	Text
note	Number
recID	Number
recVersion	Number

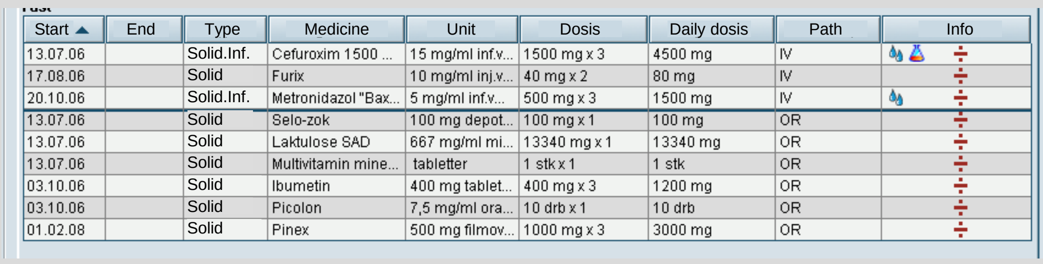
D3.3. Patient medication

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.		

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	

(Intentionally left blank)

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, system integrations, etc. 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 deals with 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 supplier must specify the price, and maybe how it depends on the complexity of the report. He might for instance use a price per Function Point (see L5-7).

Requirement 4 deals with 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), system integrations (Chapter F), etc. 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 doctor's secretary 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 in this chapter. 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.

The supplier can specify that his system automatically does this and how.

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 additional complex rules.

E3-1 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.

E3-2 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 background 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.

E3-3 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.

E3-4 shows a more complex rule as a state-transition diagram. It shows the states of a LabSys service request. The possible states are shown as boxes with round corners. It corresponds to the service states mentioned in D3-4. The diagram shows how the state of a LabSys service changes inside the EHR system.

The clinician creates the service in the EHR system, which set the state to *Ordered.* At the same time the EHR system sends a LabRequest to LabSys. LabSys sends a *LabConfirm* message to the EHR system, which sets the service state to *Confirmed*. Next the clinician sends the physical sample and marks it in the EHR system, which sets the state to *Started*. Later LabSys sends a message with the result to the EHR system, which set the state to *Completed*. When the doctor later sees the result in the EHR system, the system sets the state to *Assessed*.

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 real requirement.

E3. Business rules and complex calculations

Some business rules are specified in the task steps, e.g. *Check that* … (example in C11-6). Other business rules are 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 according to the diagram. The diagnosis disappears when the system automatically cleans up the data after 20 years.

Figure 6. Diagnosis states

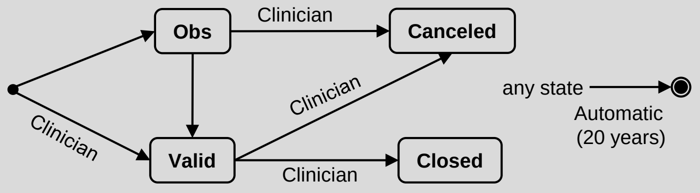
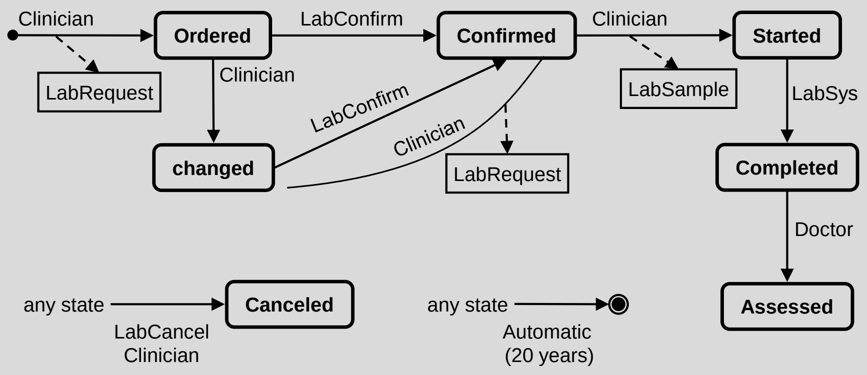


Figure 7 shows how the state of a LabSys service changes inside the system. A clinician creates a LabSys service in state *Ordered*. During the creation, the system sends a LabRequest to LabSys. When LabSys sends a *LabConfirm* message to the system, it changes the service state to *Confirmed*. A clinician 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 F10.

Notice that the template not only asks for expansion functionality, but also for the rights to use it (E4-8). 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. In this section, "customer" means the customer's own IT staff or a third party authorized by him. The customer expects to be 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 to be able to add new types of services. The requirements below state the demands.

Expansion requirements:	Example solutions:	Code:
1. The customer can define new types of services based on data 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, 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, 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's IT staff or a third party 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 the data stored in the system.		

F. Integration with external systems

Integration means that two systems shall communicate. Usually it is a matter transferring data from one system to the other. 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 (**B3**). So usually we need explicit integration requirements.

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 section A1, but 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 SKS system and LabSys. Note that he is not required to integrate with new external systems. Someone else may do it.

Which system should initiate the data transfer? 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. Access rights to data. Who is allowed to transfer what?
B. Protection of data: Avoid data loss, duplication, and corruption.
C. Documentation and rights: What to document? Who may use it for what?
D. Responsibility: Who will make and test the integration and how will the "other end" help?
E. Task support: Can the user tasks be supported well with this integration?
F. Data to import from the external system: Which data?
G. Data recency: How old is the data that the customer's system shows? This is the key concern in integration. With the ideal SOA architecture (see below), the data on the screen will be only a few seconds old. With a batch-wise transfer it may be hours or weeks old, but this may be sufficient.
H. Response time at import: When the system requests import of data, how fast should it be transferred?
I. Data to export: Which data and when?
J. Response time at export: When the system requests export of data, how fast should it be transferred?
K. Other functionality: Can the system order other functions in the external system, for instance remind users or print data? Or does it offer functions itself?

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

F. Integration with external systems

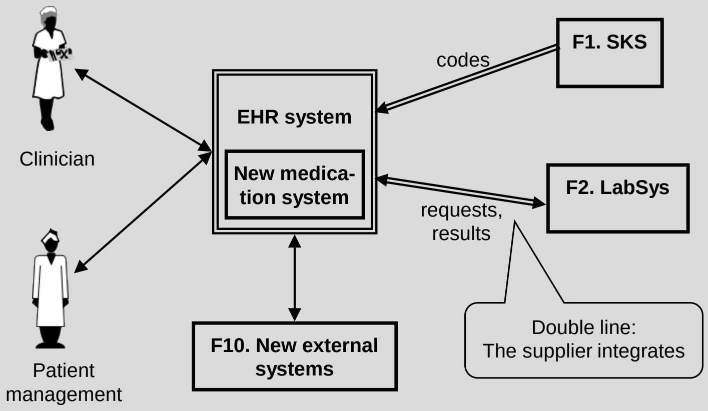
The system must integrate more or less closely with the external systems shown in Figure 8 (context diagram). The integration comprises data sharing or replication, and the ability for the user to activate functionality in the external systems.

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

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

Here is a short explanation of the external systems:
F1. SKS: The National Health Classification system. The National Health Organization updates it regularly.
F2. LabSys: The customer's present lab system for …
F3. …
F10. An external system that the customer will buy later and integrate.

Figure 8. Context diagram



Requirement note: Response times
The response times specified in this chapter must be interpreted in the same way as in L1, i.e. with L1's fractile, measured in peak load periods, etc.

Integration aspects
For each integration there are many aspects to consider:
A. Access rights to data.
B. Protection against loss of data.
C. Rights and means to integrate the system with other systems or migrate data.

D. Integration responsibility, e.g. the supplier, or the customer with support from the supplier.
E. Tasks the integration must support.
F. Data import from E (the external system). Which data to import.
G. Data recency (how old may the local copy of the data be).
H. Response time at import.
I. Data export to E. Which data to export.
J. Response time at export.
K. Other functions, e.g. warnings to the user or E.

For practical reasons the requirements in group A, B and C are written as common integration requirements, which means that they are valid for all integrations where relevant.

What should the requirements say about the external systems that the supplier has to integrate with. The systems exist and the supplier has to know about their technical interfaces (API's or XML services) in order to estimate his own integration costs. Yet the customer rarely has this information.

The customer can refer to the supplier of the external system, but often he is not willing to help. He sees the new supplier as a competitor. The customer must ensure that the old supplier will help, for instance by buying the necessary rights. This is an important assumption for the supplier. It must be stated above the requirements table, in the same way as other assumptions the supplier can make.

To avoid that the new supplier later causes similar troubles, he must accept requirements F0-6 to 9. Then the customer doesn't have to negotiate with him the next time something has to be integrated.

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 stored only in their source system. Other systems must 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, SOA may become a really expensive solution for other reasons too. The COTS system retrieves data from its own database, but now data must be retrieved through SOA from another system. 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 (batch transfer). This is usually much easier to add to a COTS system.

F0. Common integration requirements

This section covers requirements that apply for all the integrations where relevant and 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 data confidentiality doesn't depend on only special PC programs showing the permitted data. It would be too easy to install a spy program that lets the user peek at the forbidden data. This requirement could also be considered a security requirement and placed in section H1.

F0-2 to 3 require the system to protect against technical problems with lost or duplicated data. This could also be considered a security requirement and placed in section H3.

F0-4 recognizes that it may be necessary to analyze the actual data transfers, and asks for ways to do it.

F0. Common integration requirements

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

A. Access rights to data: May be moved to H1	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.		

B. Protection of data: May be moved to H3	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. The system must support error tracing at data transfers.	Logging all transfer errors.	

C. Documentation and rights:	Example solutions:	Code:
5. It must be easy to add new interfaces, e.g. SOA services, database queries, or API's.	The system provides an OData interface that allows the client to define services. Or: The supplier can do it at a fixed price.	
6. 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.		
7. 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 (see B2-4).	
8. The customer must have the right to use the documentation and the interfaces themselves.		
9. The supplier must loyally support the customer in the integration or migration effort with qualified staff at a fair price.		

F0-5 specifies that it must be easy to add new technical interfaces to the system, e.g. SOA services. Although some customers believe they can define the necessary services in the requirements, experience shows that new services are often needed. If you need a new service, it is very expensive because the suppliers of the two systems have to agree and test their systems together. An alternative is to use an OData interface (Open Data Protocol) where the client to a large extent can define on his own what he wants to retrieve (like an SQL statement).

F0-6 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.

F0-7 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 existing 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 the supplier's way of documenting will suffice for third party integration with the EHR system (see section B2).

F1. Simple one-way integration (SKS)

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

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

Tasks:	Which tasks utilize the integration?
E-support:	Who has the rights to integrate? How to get the documentation of the external interface? Who can provide support?
E- updates:	How frequently are SKS codes updated inside the SKS system?
Data volume:	How much data to transfer?

The template shows two versions of the requirements table for F1. One where we carefully have considered all points from D to K, and one where we only write the strictly necessary requirements.

All points considered

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 SKS.

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 SKS, 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 SKS or transfer data to SKS.

The short version

Here we can do with just two requirements: (1) The supplier is responsible. (2) The new SKS tables must be used by the system shortly after having been released.

F1. SKS

E-data (external data):	The SKS tables comprise codes and corresponding names for diagnoses, services, health departments, etc.
Tasks:	The codes and names are used in most of the tasks. However, the department codes are retrieved from another system.
E-support:	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-updates:	The department data are updated on a monthly basis, the other codes every three months.
Data volume:	The SKS tables comprise around 100,000 records, each around 100 characters.

Alternative 1: All points considered

D. Integration responsibility:	Example solutions:	Code:
1. The supplier must integrate the system with the SKS tables.		

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

F. Data import:	Example solutions:	Code:
2. All codes and names are needed, except the department data.		

G. Data recency:	Example solutions:	Code:
3. S-data should not be older than a week.	The system imports E-data every __ days. Or: IT support starts a transfer when the Health authorities announce that data are available.	
3p. Sometimes new SKS codes conflict with local codes or cause other problems.	IT support can roll SKS 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.	

H. Response time at import: No requirements.	Example solutions:	Code:

I. Data export: None.	Example solutions:	Code:

J. Response time at export: N/A.	Example solutions:	Code:

K. Other functions: No requirements.	Example solutions:	Code:

Alternative 2: The short version

Integration requirements:	Example solutions:	Code:
1. The supplier must integrate the system with the relevant SKS tables.		
2. The system must use the new tables shortly after their release.	The tables are imported within a week after their release.	

F2. Two-way integration (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 LabSys.

Tasks:	Which tasks utilize the integration?
E-support:	The customer refers to a technical document and promises that a specific company, MediData, can provide support (see the introduction to Chapter F). The customer has contracted the necessary rights.
E-data updates:	Each update corresponds to LabSys generating a reply.
S-data updates:	S is the EHR system. S-data are the requests. An update corresponds to sending a request.
Data volume:	A reply consists of around 500 characters per result.

F2-1 specifies that the EHR supplier has to make the integration. He can assume support from MediData as promised under E-support.

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 supplier to explain what he considers a good solution.

F2-3 specifies the data to import. The data correspond to Service records 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. They assume that the supplier can work out a solution with MediData, since electronic data at present are transferred over night.

F2-6 specifies the response time for data import (getting the test reply). The example solution allows time for LabSys to send the reply. In general the supplier will have troubles meeting a response time that includes time for external system requests. So a fair requirement allows the time needed by the external system.

As mentioned in the requirement note at the introduction to Chapter F, response times must be interpreted in the same way as in section L1, e.g. with fractiles and peak load periods.

F2-7 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 and be specified in "other functions" (F2-K).

F2-8 specifies the response time for data export (sending the request). There are actually two times involved: The time until the user can continue typing or clicking, and the time until the user can see the LabSys confirmation.

F2-9 and 10 specify that the EHR system can notify its own users and LabSys about new and missing replies.

F2. LabSys

E-data (external data): LabSys version yyy. Users can request lab tests from LabSys. The sample itself is delivered by … and the reply comes on fax and electronically. One reply may contain several results.

Tasks:	LabSys is used in connection with task C10, Perform clinical session.
E-support:	The technical interface to LabSys is described in … MediData supports LabSys in Denmark and can provide integration support. The customer has contracted the rights with MediData.
E-data updates:	LabSys generates replies continuously by fax, but at present the electronic replies are only sent as a batch over night.
S-data updates:	The entire hospital requests around 8000 tests a day, mainly between 8:00 and 16:30.
Data volume:	Each reply consists of one or more results, each of around 500 characters.

D. Integration responsibility:	Example solutions:	Code:
1. The supplier must integrate with LabSys.		

E. 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 ID.	

F. Data import:	Example solutions:	Code:
3. All E-data that can match the data in section D3.		

G. Data recency:	Example solutions:	Code:
4. S-data should not be older than 3 hours.	The system imports E-data every __ hours. Or: Data is imported at E request when they are available. Or: Data is always retrieved from E.	
5. Sometimes the latest results are needed for a specific patient, e.g. during surgery.	The system retrieves data on the user's request. Or: Data is always retrieved from E.	

H. Response time at import:	Example solutions:	Code:
6. When the user requests a lab reply, it must be so fast that the user doesn't lose patience.	The result is visible within __ s plus the time LabSys needs to send the reply. (The customer expects 3 s.)	

I. Data export:	Example solutions:	Code:
7. The user can send LabSys requests through the EHR system (S)..		

J. Response time at export:	Example solutions:	Code:
8. A lab request can be sent and the user continue typing within the mental switching time (around 1.3 s). The confirmation from LabSys should be visible a bit later.	Typing is possible within __ s. (The customer expects 1.3 s.) The confirmation from LabSys appears __ s after LabSys has sent it. (The customer expects 3 s.)	

K. Other functions:	Example solutions:	Code:
9. S can notify the user about new or missing LabSys replies.		
10. S can notify LabSys (E) about missing LabSys replies (reminders).		

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 the customer (or a third party) is able to implement such integrations. In section F10 the customer tries to get information about what kind of integrations he can make himself.

E-support explains that it is the customer's responsibility to get documentation for the external system (of course).

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 according to F0-9.

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 import from the external system: Being able to transfer data on request or periodically; transferring only data younger than a certain point in time; transferring only data about a specific patient. The customer imagines that he can configure the EHR system to do these things.

F10-7 asks for response times. However, it is not possible for the EHR supplier to promise response times unless he knows the load of the EHR system and the kind of transfer. If the customer integrates the EHR system heavily with other systems, the EHR system can become overloaded and respond slowly.

How can we make a fair requirement about this? One way is to ask for additional capacity so that the EHR system can carry a load x times as high as the load specified in L1, and still provide the response times specified in L1. The customer can then use the additional capacity for data transfers. This is what F10-7 asks for.

F10-7p mentions a known problem in this kind of integrations: An unusually long data transfer may block the system and ordinary small transfers.

F10-8 to 11 are similar to F10-3 to 6, but specify features for data export to the external system.

F10-12 and 13 ask for a list of the functionalities the EHR system can use in an external system and a list of those it offers to external systems.

F10. Integration with new external systems

The customer expects that he can integrate new external systems with S - with little or no help from the supplier of S. This section lists the demands such integrations might have and asks for the supplier's suggestion for what he can deliver to meet the needs.

External system:	In principle any system. Examples: X-ray system, mobile applications, specialist system for intensive care.
Tasks:	Defined later.
E-support:	The customer's responsibility.
E-data updates:	Defined later.
Data volume:	Defined later.

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

E. Task support:	Example solutions:	Code:
2. For mobile applications E may in some periods be off-line. When E connects to S again, data synchronization is needed.	The customer can configure S to automatically synchronize data at reconnect.	

F+G. Data import and data recency:	Example solutions:	Code:
3. S can import data from E assuming that they fit into S's existing data tables.	The customer can configure S to import at S's request or E's request.	
4. S can periodically import data from E.	The customer can configure S to do this.	
5. S can optimize the import by asking only for data younger than a certain point in time.	The customer can configure S to do this.	
6. S can optimize the import by asking for data about a specific patient only.		

H+J. Response time at import and export:	Example solutions:	Code:
7. S can scale up to carry a significantly higher load than specified in L1 with the response times specified in L1. The customer may use this additional load for data transfers.	The system can scale up to handle a load ___ times as high as required in L1. (The customer expects 2 times.)	
7p. When a long transfer is in progress, it may block for shorter transfers so that they have a very long response time.	The system can handle several concurrent transfers.	

I. Data export:	Example solutions:	Code:
8. S can export data to E assuming that the data exist in S's existing data tables.	The customer can configure S to export at S's request or E's request.	
9. S can periodically export data to E.	The customer can configure S to do this.	
10. S can optimize the export by sending only data younger than a certain point in time.	The customer can configure S to do this.	
11. S can optimize the export by sending data about a specific patient only.		

K. Other functions:	Example solutions:	Code:
12. S can use functionality in E, e.g. request services or warn about missing requests.	The supplier is asked to specify the functionality S can use.	
13. 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.	

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 in other chapters.

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 serve on the existing equipment. "Serve" means meeting the response time, availability and storage requirements of Chapter L.

G1-2 asks the supplier to specify any additional equipment needed to handle the full nominal load.

Often some parts of a system are executed in an internet browser, e.g. parts intended for the public. G1-3 requires that these parts can execute on common browsers. The solution column lists the browsers the customer considers.

Many IT gurus claim that everything should be web-based, in order that it can be used everywhere. Unfortunately this is not correct. Simple web pages, okay, but when things get complex, they are browser dependent. Short-cut keys, database connections and security settings vary from browser to browser. In practice the supplier must include tests in the program to see whether things must be done one way or another. And it has to be tested on all browsers - also when a new browser version is released.

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.

Here too we need requirements for browser support.

G3. The supplier operates the system

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

Here too we need requirements for browser support.

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. Within any 1 second period, the optical fiber net leaves 50% of the capacity for the EHR system.
7. No other applications run on a PC when it runs 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 ____.	
3. The browser-based parts must be able to run on common browsers.	MS-Internet Explorer, Chrome, Safari	

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.		
4. The browser-based parts must be able to run on common browsers.	MS-Internet Explorer, Chrome, Safari	

G3. The supplier operates the system Alternative 3: Supplier's problem

Platform requirements:	Example solutions:	Code:
1. The supplier operates the system and uses the necessary equipment to meet L1, L2 and L3.		
2. The browser-based parts must be able to run on common browsers.	MS-Internet Explorer, Chrome, Safari	

H. Security

The purpose of security requirements is to guard the security factors (CIA+A): Confidentiality of data, Integrity (correctness) of data, Availability (of data and processor capacity), and Authenticity of the users (that the user actually is the person he claims to be).

H1. Login and access rights for users

This section describes the situations where the user's access rights must be checked. The system must guard Confidentiality, Integrity and Authenticity. 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 or more convenient security.

Alternative 1: Login as today

H1-1 says that the user must be identified with the existing method and what this method is.

H1-2 says that access is only allowed to users with the proper rights. The example solution mentions two ways to do it.

Alternative 2: Better security wanted

H1-1 again says that the user must be identified. The example solution suggests the traditional approach but also an alternative identification.

This requirement doesn't say anything about the length of passwords. The length is considered a protection against intruders and is handled in section H5-3.

H1-2 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.

H1-3 says that the rights must be checked and mentions the existing problem with a password for each system. A 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.)

H1-4 mentions a threat to protect for, e.g. by changing passwords.

Possible access rights and their granularity

For alternative 1 as well as 2, it is important to specify the possible access rights. They are shown as a requirement note below the requirements table. In the EHR system there are separate rights for prescribing drugs and seeing patient data. 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. Notice that a person can have multiple rights.

Many customers neglect the list of rights 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.

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:	Example solutions:	Code:
1. Identify the user with 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 and more convenient security wanted

Subtasks for user access:	Example solutions:	Code:
1. Identify the user. (See section H5-3 about the length of passwords.)	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 and data. (See the requirement note below.)	The database system checks the rights. Or: The user screens show only the functions and data he is allowed to use.	
3p. Problem: Today the users have a password for each system. It is cumbersome to switch between systems 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).	
4. Stolen passwords are often traded by criminals. Limit the possibility.	Users must change passwords regularly. When a leak has been detected, it must be possible to reset all passwords.	

Requirement note: Possible 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 physician in department M might for instance have rights 1, 2, and 3, while a supervising physician 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. This is specified as an assumption before the table.

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. They are example solutions and not requirements.

The template shows two alternatives. Alternative 1 expects that the new system includes functionality for creating users, changing rights, etc.

Alternative 2 expects that security rules are handled by the customer's existing security management. The EHR system should ask the existing system when checking user passwords and rights.

H2. Security management

Each department has its own security management.
Or: Security management is centralized for the entire hospital.
The work in security management includes the following subtasks.

Alternative 1: The new system has its own security management

Subtasks for security management:	Example solutions:	Code:
1. Assign or remove rights for a user.		
1a. First, create the user.		
1p. 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.	
1q. Problem: A temporary employee has been appointed in a hurry and is not yet in the personnel system. Needs access rights anyway.	Possibility for temporary registration in the department, bypassing the central department.	
1r. 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. physician in department M and supervising in department N. Each role has one or more rights, e.g. prescription and diagnosing.	
1s. 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.	
2. Create new roles with new combinations of rights.		
3. Get an overview of who has which rights and whether some rights have not been assigned to anyone.		

Alternative 2: Use the existing security management

The customer uses LDAP and AD and wants to manage all rights in this way.

Subtasks for security management:	Example solutions:	Code:
1. Create and remove users.	Leave it to the existing security management.	
2. Assign or remove rights for a user.	Leave it to the existing security management.	
3. Check that the user has the necessary rights. (Strictly speaking, this is a subtask in H1).	The EHR system retrieves the rights data from the customer's existing system.	

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.

These requirements guard Availability and Integrity of data.

With the help of a security expert, the customer may ask for protection against many other sources of data loss. The template shows an example where the supplier let a subcontractor operate the system (in the cloud). The subcontractor didn't store data properly. As an example, he stored the backup version of the database at the same disk as the primary database. The day when the disk collapsed, database as well as backup disappeared.

In the template this experience is treated as a threat similar to other threats. The customer has suggested some solutions.

H4. Protection against unintended user actions

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

H4-1 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 a way the customer could be convinced.

H4-2 and 3 specify protections against simple mistakes and use of undo at unexpected system response.

H4-4 recognizes that not all functions are undoable, but asks for ways to prevent that they are used by mistake.

H4-5 asks for a way to stop a function that turns out to take a long time.

These requirements guard Integrity of data, and for H4-5 also Availability.

H3. Protection against data loss

Data may unintentionally be lost or misinterpreted in many ways.

The system must protect against:	Example solutions:	Code:
1. (See F0-2 for protection of data against loss or replication during transfer between systems.)		
2. (See F0-3 for protection against concurrency problems with external systems.)		
3. Local concurrency problems, for instance that user A makes a prescription, but before the system has recorded it, user B makes a prescription that interacts. Neither A nor B will notice the conflict.		
4. Disk crash	Periodic backup or RAID disks.	
5. Fire and sabotage	Remote backup at least 10 km away …	
5p. The system operator doesn't store the data properly, as an example stores the backup data on the same drive as the database. Mainly observed with subcontractors.	The main contractor regularly audits whether it is done properly. Or: The customer gets a weekly backup of all his data for his own storage.	

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.	May be hard to test at delivery, but the supplier's issue log and a description of the supplier's test methods may help.	
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. Prevent mistaken use of undo-able functions.	Position the button so that it is not hit accidentally - or ask for confirmation.	
5. The user must be able to interrupt functions that take a long time, e.g. a long data transfer, without compromising data integrity.		

H5. Protection against threats

This section deals with threats caused by viruses, hacking, SQL injection, Trojan Horses, etc. They can threaten all the security factors (Confidentiality, etc.). In order to identify the most important ones, somebody should make a security risk assessment.

During 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 deal with the most serious threats.

In practice, protection against threats is the weakest part of security requirements, and proper security risk assessments are rarely made. Customer as well as supplier believes that following standards is sufficient (e.g. H5-6). To make things worse, the list of potential threats keeps growing as attackers become smarter.

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 of threats.

We often see security requirements that specify a solution rather 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 possible to break eight-character passwords in around a month*.

H5-3 handles this as a threat. We can now see that there are other solutions. The solution column mentions two that are far more convenient.

H5-5, *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 all the threats in this area and includes the serious ones as requirements in H5.

H5-6 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.

H5-6 may be a useful addition to the security risk assessment and the specific threat requirements (H5-1 to 4 in the example). However, it should not be considered a replacement for the risk assessment and the specific threat requirements.

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 the customer's profile. This is costly to both parties, so it should be made during the project, maybe during the early proof-of-concept (B3).

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 and Caps as well as … (cumbersome). Or: at least 5 seconds between login attempts. Or: Block access after 3 attempts.	
4. SQL injection (the intruder types a database command where the system expects e.g. a person name; as a result the system carries out the database command).		
5. DoS attack (Denial of Service). An attacker sends so many requests to the system that it is paralyzed.		
6. Unauthorized persons getting access to personal data. Too open-ended, see the guide booklet.		
7. 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.		
8. The supplier must follow developments in the security area and deliver safeguards.		
. . .		

Alternative 2:

The customer has not made a security risk assessment.

Threat protection:	Example solutions:	Code:
1. The supplier must list the threats that are most serious for this kind of system and specify 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 lack of 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, designers and expert users believe they can scrutinize the screens and see whether the system has adequate usability. 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 that we observe users carry out some realistic tasks by means of the system or a primitive prototype of it. We log events where the user needs help, 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 explains the details as a requirement note below the table. 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 remove 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

The template suggests two alternatives for usability requirements.

Alternative 1: The system has a user interface already

Often the new system is almost finished and it is little you can change in the user interface. If the customer complains about the cumbersomeness of the interface, he is told that it is a COTS system that cannot be changed, or that he has seen it before he bought it (implying that the users simply are too stupid). However, he is willing to repair serious errors, such as the database crashing.

Alternative 1 has only one requirement: Critical usability problems are to be treated as other errors, i.e. being prioritized and repaired according to how serious they are for the customer (see L5).

I. Usability and design

I1. Ease-of-learning and task efficiency

Alternative 1: The user interface exists already

Although the system has a finished user interface, it may in some places give the users considerable troubles. The customer wants to avoid the situation where the supplier rejects the problem with reference to the system being a COTS system.

Requirements for handling usability problems:	Example solutions:	Code:
1. Critical usability problems (see definition in the requirement note below) must be handled as system errors in the same way as other errors in the system.	The error is handled by the support organization and eventually transferred to maintenance.	

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.

Requirement note: Test tasks
A good test task corresponds to something a real user would have to do. It must be presented in such a way that it doesn't guide the user. Here is a good and a bad example:

Test task 1 (good): Prescribe medicine: The patient complains about pain. Use the system to treat the problem.
(When the user carries out the task, notice whether he checks the existing medication situation before he prescribes something.)

Test task 2 (bad - guides the user): Prescribe medicine: The patient complains about pain. Enter the patient ID and choose the medication screen. Look at the other medications and decide what to prescribe. Close the medication screen and select the prescription screen …

Alternative 2: Essential parts of the user interface will be developed

I1-1 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. We also define the detailed usability requirements to be used later. At the same time we design parts of the user interface and test it for usability. 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 I1-1 in the template and kept I1-2 to 6. We would thus require that users were able to carry out all 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 I1-1 is to find some reasonable usability requirements early in the project.

I1-2 to 6 are outlines of usability requirements that have to be defined in detail during the proof of concept. For instance the precise test tasks have to be defined and the numbers in column 2 must be filled in.

I1-2 checks that after the planned introduction, users can carry out their tasks with minimal support from others.

I1-3 checks that error messages are usable. Why is this necessary when we have checked that users can carry out their tasks? Because users only encounter a few error messages during the test tasks. I1-3 makes it possible to test more messages, also those that rarely occur.

I1-4 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 is irrelevant, of course.

I1-5 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 then train other users. One way is to provide courses for the super users. In J2-1 we ask the supplier to run such courses.

I1-6 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 used occasionally

The template shows requirements suited for production systems that are used on a daily basis. However, I1-2, 5 and 6 are usually irrelevant for websites used occasionally by the public. There are no super users around, and efficiency is unimportant.

Test tasks

The basic idea in section I1 is to do usability testing to detect and remove usability problems. A crucial part of this is how you define the test tasks that users will carry out. The template suggests that you write some test tasks in a requirement note before asking suppliers for proposals. This will give the supplier an idea what you ask for. The parties can revise the test tasks during the early proof of concept.

The template gives an example of a good test task and one with "hidden help", but there are other things to consider, for instance how well the test tasks cover the most important aspects of the system. See for instance Lauesen (2005), Chapter 13.

Alternative 2: Essential parts of the user interface have to be developed

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 for instance be a detailed specification of the test tasks and the numbers to be used in column 2 below.

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 5 services for a patient, e.g. lab test, scanning …	A typical user is able to order these services in __ minutes.	

I2. Accessibility and Look-and-Feel

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

I2-1 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.

I2-2 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 (and many other purposes). 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.

Some requirements specifications replace I2-2 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, but they cannot be verified by a computer.

I2-3 is an example where the language must be specified.

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 for 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.		

J. Other requirements and deliverables

This chapter contains requirements that don't fit into the other chapters.

J1. Other standards to obey

Most required standards belong to other chapters, for instance security and usability. 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.

J2-1 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).

J2-2 specifies similar requirements for training the customer's IT staff.

J2-3 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 auditor approval or 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. 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.

J3-1 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.

J3-3 specifies in the same way the documentation for the customer's IT staff.

J3-4 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 big issue 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 for third-party development, sufficient documentation for further 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 old EHR 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. …		

J6. Testing the system

The supplier must do system testing himself (see Chapter 5), so in principle the customer need not care about it. However, experience shows that many suppliers are very bad at testing, so there is a good reason to look for what they do. It is particularly important to be able to retest the system after changes (regression testing).

J6-1 and J6-2 specify the needs and outline some solutions.

In addition the customer needs to do his own testing, e.g. the deployment test in connection with the acceptance test (Chapter 5). Many customers have been persuaded to test on a system that has been put into operation with real users and real data. This can leave strange data in the database and disturb the operation in other ways. Should be avoided.

J6-3 to 5 specify the needs and outline some solutions.

J7. Phasing out

At some point in the future, the customer wants to phase out the old system and migrate to a new one. Then new problems turn up. Which data are in the old system? How can we convert them to the new system? Will the supplier help us - also when we want to get rid of him?

F0-5 to 9 cover some of the needs, e.g. documentation of data. J7 adds requirements for the supplier's assistance and the tools needed by the customer.

It is important to have these agreements in place while the parties are on good terms.

J6. Testing the system

Requirements for the supplier's test:	Example solutions:	Code:
1. The customer wants to audit which tests the supplier makes and how well they cover.	The supplier makes his test cases and test methods available to the customer.	
2. There is a need for repeating large parts of the tests after changes.	The supplier uses regression testing.	

Requirements for the customer's own testing:	Example solutions:	Code:
3. The customer needs to test the system before accepting the delivery.	The supplier makes a test version available to the customer.	
4. Special situations must be tested.	The customer can insert special test data.	
5. There is also a need for testing with realistic data.	The supplier converts parts of the customer's existing data and inserts them in the test version.	

J7. Phasing out

In this section "customer" means the customer's own staff or third party authorized by the customer.

Requirements:	Example solutions:	Code:
1. On request, the supplier must extract all data described in Chapter D in a format that is suited for import in other systems.		
2. The customer must be able to extract all data described in Chapter D in a format that is suited for import in other systems.		
3. The supplier must loyally assist with phasing out the system and transferring it to another supplier.		
4. The supplier must carry out the work at a fair price that covers time and material.		

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 acquisitions, system integration is a big issue because the supplier of the external system to be integrated must help. K-11 specifies that the customer must provide the necessary rights, for instance buy them from the supplier of the external system. This should be stated in Chapter F as assumptions the supplier can make, but ensure it is somewhere.

In many contracts, this Chapter K 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.		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
11. Rights to integrate with the systems mentioned in Chapter F and get the support mentioned.		

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 specifies the **nominal load** of the system. The nominal load is the number of transactions the system must be able to handle per second with the specified response times. The actual number of transactions per second should be well below the nominal load. If the actual number of transactions per second is larger than the nominal load, the system need not respond as specified.

Based on the nominal load, the supplier can estimate the necessary hardware.

In the example, the nominal load is specified as the number of transactions of various kinds. Experience shows that this often creates conflicts late in the project, because transactions are of many sizes. As an example, the supplier assumed that all transactions were quite small, but in reality some of them are huge, but rare. The customer insists on measuring on these too. The advice is to specify the transactions in the nominal load more precisely, maybe during the early proof of concept.

The system is expected to be most busy in certain periods, the **peak load** periods. They are not important for the requirements, because the system must be able to handle the nominal load in any period, but the customer wants to measure the actual load and response time in these peak load periods.

The solution note describes a way to measure the response times in practice. This is not a requirement, and the supplier can specify his way of measuring the response times in L1-2.

L. Operation, support, and maintenance

This chapter specifies the supplier's responsibilities after delivery of the system itself. The requirements can only partly be verified (tested) at the deployment test. The full verification takes place later, at the operational test. Some of the requirements are only relevant when the supplier is operating the system, others only when he has support responsibility, etc. See the guide booklet.

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 …

When the system is operating, it must be able handle the number of transactions specified below with the specified response time. The figures are estimated from task frequency (Chapter C), data volumes (Chapter D) and statistics from the present operation about peak load periods. The figures are the **nominal** load, i.e. the supplier is not responsible for response time if the actual load exceeds the nominal load.

Nominal load

1. Simple queries in clinical sessions (C10): 10 per second on average.
2. Updates in clinical sessions (C10): 2 per second on average.
3. Simple queries in patient management (C1 to C4): 3 per second on average.
4. Public web access: 5 page loads per second on average.
5. …

Solution note: 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 actual load is within the nominal load above.

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 1 MB connection with low traffic on the route to the servers, but with peak load of the servers themselves.

L1-1 specifies that the required response times must be valid for a certain **fractile** of the cases. The example solution says that the customer expects 98%, but the supplier can specify another fractile. We could also expect 99%. Why not ask for 100% of the cases? Because it is unrealistic in a multi-user system. Transactions arrive randomly, and by coincidence, a lot may arrive within the same second. In this case the last ones get a very long response time. Although this is very rare, we cannot guarantee a good response time in 100% of the cases. See more in Lauesen (2002), section 6.5.

L1-2 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.

L1-3 to 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, 1.3 seconds to switch screen will not slow down the user. This principle applies to L1-4, 5 and 6.

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. L1-7 is an example where 20 seconds are acceptable.

Finally there may be functions where we for technical reasons expect response times above the ideal. L1-8 and 9 (login) are examples of this. Ideally, login should take place within 1.3 seconds, but present experience shows that we might have to accept a slower response.

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 used occasionally

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 might be unnecessary and costly.

Response time requirements:	Example solutions:	Code:
1. **Fractile.** The times specified below must apply in almost all cases.	In any one-hour period, __% of the response times must be within the limits. (The customer expects 98%.)	
2. 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.	
3. 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.)	
4. When moving from one screen to the next, data must be visible and typing possible within the mental switching time (around 1.3 s).	Data is visible and typing possible within ___ s. (The customer expects 1.3 s.)	
5. 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.)	
6. Reports used frequently must be visible within the mental switching time.	The report must be visible within ___ s. (The customer expects 1.3 s.)	
7. Reports used occasionally must be visible before the user loses patience.	The report must be visible within ___ s. (The customer expects 20 s.)	
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 name and password. (The customer expects 10 s or better.)	
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.)	

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 solution note suggests one way to calculate a breakdown period: A breakdown is always calculated as at least 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.

L2-1 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.

L2-2 and 3 state the required availability in two different operational periods. Take care not to ask for too much. It may be very expensive. As an example, operating a large system with 99% availability may cost $1 million a year, while 99.8% may cost $4 million a year. Is it worth it? An availability of 99% in the normal work hours means that the system may be out of operation 16 hours a year in these hours. An availability of 99.8% means 3.2 hours a year.

Notice the way L2-2 and L2-3 are stated. It allows the supplier to propose other figures than the customer's. See more in section A2.

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 is also 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 99%)	

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.		
2. MR scans and … are only kept for 60 days.		
3. The system must give access to archived data for the last 20 years with response times as for occasionally used reports (L1-7).		

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 as an assumption 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.

L4-1 specifies that the required response times for hotline must be valid for a certain **fractile** of the cases. The example solution says that the customer expects 95%. 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.

L4-3 and 4 specify in which periods 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.

L4-4 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 L4-4 only asks the supporter to spend a few minutes on the spot. Whether the support quality is adequate in general is hard to measure. L4-11 suggests that the parties discuss this at regular meetings.

L4-5 specifies that for indirect contacts the user must get a first reply within a few hours.

L4-6 to 7 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. In many projects there is no need at all for this, since the customer does it himself already.

The requirement note after the table explains what it means to handle a request for help (an *incident* in ITIL terminology). It is described as a list of optional 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 in the support organization. The user may often get a first reply and later additional replies as supporters investigate the case.

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 operational organization. A "supporter" means a qualified supplier employee. The support covers all hardware and software delivered under this contract.

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.

Support requirements:	Example solutions:	Code:
1. **Fractile.** The response times specified below must apply in almost all cases.	__% of the response times must be within the limits. (The customer expects 95%.)	
2. The supplier must handle user requests for help. See the requirement note below.		
2p. Problem: Even super users find it hard to 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.	
3. 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.)	
4. 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.	
5. Indirect contact: Requests sent by email, sent by web, or escalated from the direct contact. The user gets a reply within a few hours.	The supplier replies within __ work hours (8:00 to 18:00 on weekdays). (The customer expects 3 hours.)	
6. The supplier sends a supporter when this is necessary to remedy the problem.		
7. The supplier can perform remote diagnostics to remedy the problem.		
8. The supplier monitors request handling to see that requests are closed and response times met.		
9. 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.	
10. The supplier monitors the operation in order to foresee availability problems, and changes the technical configuration so that availability is maintained.		
11. Customer and supplier meet regularly to review response times and discuss prevention of problems.	The parties meet every __ month. (The customer expects monthly meetings.)	

Requirement note: Handle a request

When 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.

L5. Maintenance

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

L5-1 specifies that the required response times must be valid for a certain **fractile** of the cases. The example solution says that the customer expects 95%.

L5-2 makes the supplier responsible for keeping a log of the maintenance requests.

L5-4 says that business-critical errors must be handled quickly, e.g. within 24 hours. But 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 ordinary users tend to consider everything urgent. On the other hand, the supplier prefers to deny that it is urgent.

L5-3 suggests that the supplier decides and that his decisions are reviewed regularly (L5-5). Alternative 1 is that the local super user decides and alternative 2 that the customer's IT department decides.

When the system is to be modified or expanded, the supplier has a de-facto monopoly and can charge the customer accordingly. L5-7 shows a way around it: The size of the change is estimated as a number of *Function Points*, and the supplier has specified a fixed price per Function Point.

Function Points (FP) are a technology-independent way to measure the size of a development project. It is based on experience data from thousands of projects all over the world.

The measurement can for instance be based on the number of classes in the E/R model and their complexity, plus the number of user screens and their complexity. FP experts can use tasks to give reasonable estimates of the number of screens. A medium complex class requires 10 FP and a medium complex screen requires also 10 FP. In addition there is an adjustment factor of 0.3 to 1.6 for the project organization, etc. Changes to a system can be estimated in a similar way.

Without something like E/R and task/use cases you cannot estimate the project size.

Depending on the supplier's skills and technology, he can quote a higher or lower price per Function Point. A typical price for a FP in Denmark is 2,000 to 4,000 USD.

Expertise is needed to estimate Function Points. FP experts claim they agree very precisely when they independently estimate the same project. In case the parties cannot agree on the number of FP, you can have your local FP group decide. L5-8 might specify that this group must be used to resolve conflicts.

COSMIC points are similar to Function Points. They are much easier to use, but don't have the same extensive experience base.

L5. Maintenance

Maintenance includes defect removal, system updates and system changes.

Requirements for defect removal:	Example solutions:	Code:
1. **Fractile.** The response times specified below must apply in almost all cases.	__% of the response times must be within the limits. (The customer expects 95%.)	
2. The supplier keeps a log of reported defects as well as change requests.		
3. 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.)	
4. Business-critical defects are removed quickly.	Business-critical defects are removed within __hours. (The customer expects 24 hours.)	
5. 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 __ months. (The customer expects monthly meetings.)	

Requirements for system improvement:	Example solutions:	Code:
6. The supplier installs new versions and releases of the delivered software without unduly delay.	Installation takes place within ___ days after release of the new version. (The customer expects 30 days.)	
7. Within a period of 3 years, the supplier must offer changes at a fixed price per Function Point.	The price per Function Point is ____.	
8. Disagreement on the Function 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 measuring the size of IT projects. 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/

International Function Point Users Group IFPUG. http://www.ifpug.org/
The traditional method for measuring the size of IT projects. It is based on experience from thousands of projects all over the world.

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. In total it explains around 100 techniques with realistic examples. There is also advice on how to verify requirements (check that they are met) and how you as a supplier can convince the customer. 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

Lauesen, Soren & Kuhail, Mohammad (2012): Task descriptions versus use cases. In Requirements Engineering (a Springer Journal): ISSN 0947-3602 Requirements Eng (2012) 17:3-18, DOI 10.1007/s00766-011-0140-1. Shows with experimental results why use cases aren't suited for requirements and

how the task approach solves the problems. See also:
http://www.itu.dk/people/slauesen/SorenReqs.html#UseCases

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.