Requirements Problems in the Development of a New User Interface for Healthcare Equipment

Maria Holmegaard, Jens Bæk Jørgensen, Michael Sørensen Loft, Martin Stig Stissing
Mjølner Informatics A/S
Aarhus, Denmark
{mho,jbj,mls,mss}@mjolner.dk

***Abstract*—In August 2013, our company started work for an industrial customer. First, we developed a prototype and conducted field studies in small-scale projects. This was successful and the basis for a larger project about development of a new user interface for healthcare equipment. A main aim for us was to use this project as starting point for establishing a strategic, long-term relationship with this customer. However, we were not successful. In November 2014, our customer chose to take over the development themselves. We were too expensive, used too many hours and were not able to provide useful estimates, they said. In this paper, we describe the project and analyze causes to our customer's decision. We also look at possible alternatives to the actions we took in the project and discuss whether we could have done better. A root cause to our customer's dissatisfaction is related to requirements and handling of requirements.**

***Index Terms*—Pragmatic requirements engineering; collaboration with stakeholders; requirements engineering for user experience including ethnography, design and usability.**

I. INTRODUCTION

Our company, Mjølner Informatics, develop custom-made software solutions for Danish and international customers. We are around 80 employees and have expertise in development of a broad range of system types. Our services span all software engineering activities, including domain and user research, requirements engineering, software architecture, interaction design, graphical design, implementation and testing.

In this paper, we consider a recent project for a particular industrial customer with headquarters in Northern Europe. One of our main aims with the project was to use it as starting point for establishing a strategic, long-term relationship with this customer. However, the outcome was not as we had hoped. Our customer were not satisfied. The main reason is that we were seen as being too expensive because we used more hours than the customer found reasonable. The project frame was a time and material agreement, and was based on an estimate that was made prior to the development project and before the agreement was signed.

In the agreement about the development project, it was said that "During the project, the assumptions for the project proposal may change and this may result in changes to the budget". This quote reflects that there were a number of known uncertainties at the time when the project was initiated.

When our current engagement with our customer ended, in November 2014, we had used approximately three times the number of hours that was originally estimated for the development project. Based on this information alone, it is not surprising that our customer were not entirely satisfied.

However, there are reasons to this outcome. Some of the reasons are general project management issues including that a number of necessary deliveries from our customer were delayed and that the timeframe for the development project doubled. Initially, it was agreed to be March – June 2014, but the actual timeframe was March – November 2014. Other reasons are clearly related to requirements and requirements handling. Examples are that the product increased in size – around twice as much functionality than originally agreed - and that requirements specifications for a number of components with which the product should coexist and communicate were either missing or incomplete.

Lauesen [5] classifies requirements in goal-level, domain-level, product-level, and design-level requirements. We have encountered problems at all four levels, as we will discuss.

Our goal with this paper is to describe and analyze the project with focus on the requirements related problems in order to get a better understanding that we can benefit from in future projects with similar characteristics. For confidentiality reasons, we keep the customer anonymous.

The product under discussion is a graphical user interface for healthcare equipment, used at hospitals and nursing homes. First, we developed a prototype and conducted field studies, with a Mjølner project team which was quite small; we will describe the details later. In the subsequent development of the full product, Mjølner's project team was organized with a project manager, a software architect, a user experience specialist (requirements engineer), a digital designer and four developers; the authors of this paper together represent all roles, except the digital designer.

The structure of this paper is: Section II presents the project and its timeline. In Section III, we outline the product itself. In Section IV, we describe the requirements specification. In the central Section V, we list and discuss a number of requirements related problems that we encountered during the project and we discuss alternative courses of actions that we could have taken. Section VI addresses how the requirements problems contributed to increased estimates. The conclusions are drawn in Section VII, which also includes a discussion of related work.

II. The Project and its Timeline

The project was a sequence with three major activities, which we will describe below.

A. Initial Prototype and First Version of Design-Level Requirements, Based on Given Product-Level Requirements

In August 2013, we started developing a prototype for our customer. The overall goal of the prototype was to use it as an instrument to maintain or even strengthen the customer's leading position in the market. The prototype was a tangible indication to the market showing that soon the customer would launch an exciting and innovative new product. Thus, the overall goal-level requirement (for both the prototype and the full product) was, as we saw it, to achieve a strengthened market position for our customer.

The prototype was presented as a showcase for our customer's customers (our customer's equipment is built into equipment manufactured by other companies) to find out the demands from the market and as a way of testing the new concept and design of the user interface. The prototype was demonstrated at a major international exhibition in November 2013 and was successful. Our customer's customers found the prototype very interesting and promising.

The customer orally delivered the product-level requirements for the prototype to Mjølner. The requirements were based on our customer's own experiences and collected from their other products. They were to a large extent gathered from local sales people, who had been in contact with customers in a number of different countries.

Through the prototype, we provided the first version of the design-level requirements in the form of the interaction design [9] and graphical design of the user interface.

The prototyping activity was rather small for us. From our side, it involved only the user experience specialist and two developers. It used well-known hardware and communication protocols. The customer were very satisfied with our work.

B. Validation and Elicitation at Hospitals and Nursing Homes, and Alignment of Design-Level and Domain-Level Requirements

As a way of validating the requirements implemented in the prototype and doing further requirements elicitation, we visited a hospital and a nursing home in the customer's local market. Here, we did elicitation through field studies, where we observed healthcare personnel doing their daily work; we also conducted some interviews. Moreover, we validated the prototype by carrying out a number of small tests.

We acquired a larger understanding of the domain. We got insight into the different use scenarios that the new product was going to be a part of. The healthcare personnel, who would be future end users of the product, were met in their work environment to get a more realistic experience of the use scenarios. We observed their varying work tasks, which were characterized by routines with a high level of efficiency.

After the observations and interviews, we conducted a validation of the current prototype in order to improve the design of the user interface, before it was determined as the final concept the new product should be based on.

To let users test the prototype in relation to the actual use scenarios gave valuable information that could be taken into account in the development of the product. Furthermore, it was a crucial element in our customer's sales communication that the concept actually was tested in, and optimized for, real use situations based on input from tests by users.

With the results from the field studies and interviews, we were able to extract concrete domain-level, which later in the process were used to enrich the design-level requirements for the new product. In essence, we aligned design-level requirements with domain-level requirements.

Again, our customer were fully satisfied with our work on this activity, which had the user experience specialist as the only Mjølner employee involved.

C. Development of a Full Product and Continued Work with Product-Level and Design-Level Requirements

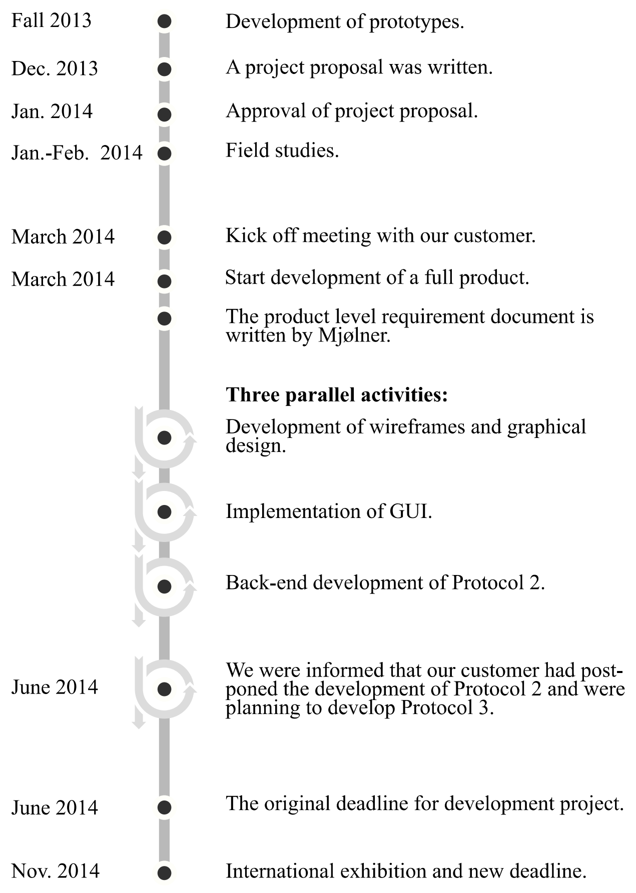
The agreement to develop a full product was signed in January 2014 and was based on a project proposal that was written just before Christmas 2013. The proposal was at a very overall level and left many details to be decided and specified later. The project proposal assumed that the product should be developed by improving and extending the latest prototype and use known and existing communication protocols that had already been used by us together with the prototype.

The product would be developed by Mjølner and the customer in collaboration.

When we started the development of the full product, in March 2014, we had a kick-off meeting with our customer. Obviously, a main purpose was to meet the people from the customer that we were going to cooperate with. We met the newly appointed head of software development, the project manager, a protocol developer and an accessories developer.

In addition to these people, key stakeholders from the customer were the head of development (not to be confused with the head of software development; these two are different people), the head of sales and two sales people. None of these participated in the kick-off meeting.

It was the head of development, who had signed the contract about the full product development, and it was the head of sales who had signed the contract about what was called design (and which is more properly termed "requirements engineering and design"). This meant that our project from early 2014 was regulated by two separate contracts.



Fall 2013 — Development of prototypes.
Dec. 2013 — A project proposal was written.
Jan. 2014 — Approval of project proposal.
Jan.-Feb. 2014 — Field studies.
March 2014 — Kick off meeting with our customer.
March 2014 — Start development of a full product.
The product level requirement document is written by Mjølner.
Three parallel activities:
Development of wireframes and graphical design.
Implementation of GUI.
Back-end development of Protocol 2.
June 2014 — We were informed that our customer had postponed the development of Protocol 2 and were planning to develop Protocol 3.
June 2014 — The original deadline for development project.
Nov. 2014 — International exhibition and new deadline.

Fig. 1. Timeline.

A hard deadline for the project was mid November 2014, for this year's main international exhibition. Development and continuation of interaction design and graphical design took place throughout most of the project. The specification of the user interface was done in collaboration between the sales department at our customer and the user experience specialist from Mjølner. We experienced that the scope increased several times, often because the sales people had new input from the global market that they wanted us to incorporate.

During the development of the full product, we sent a short status report to the customer each week. The status report contained information about how many hours we had used until now and estimates for remaining work, when possible. The report explicitly distinguished between, on one hand, hours used for requirements and design and, on the other hand, hours used for development.

Our customer became increasingly dissatisfied with us. In early November, immediately before the exhibition, the product was almost complete, and it was in this way ready to be presented at the exhibition. It would have been possible for us to complete the development shortly after the exhibition, but the customer chose to take over the remaining development themselves.

Figure 1 gives an overview of the project's timeline; Protocol 2 and Protocol 3 are mentioned in the figure and will be discussed in Section V.E.

We encountered a number of problems in the development of the full product. We will discuss requirements-related causes to these problems in Section V. However, in the next two sections, we will provide some more background by describing the product itself and the requirements specification in more detail.

III. The Product

The product unifies, encompasses and replaces a collection of the customer's existing control panels and is the first to have a graphical user and touch interface.

The product is adaptable in the sense that it accommodates the multitude of variations of setups of equipment and accessories. A piece of equipment might or might not have motors adjusting the position of various constituents, sensors to monitor certain situations etc.; the product will adapt to these different setups.

Variations of the product to suit specific customers of our customer are a significant part of the overall product strategy. This entails skinning the graphical user interface and supporting additional customer specific accessories.

The product is a piece of software suited for custom embedded hardware. The hardware is to be developed by the customer. The product is to be sold in many copies, so there is a strong focus on keeping the unit cost low. This implies that the product is based on hardware with limited processing power and memory.

The programming language used is C/C++ as this is suitable for both the graphical framework used and for integrating with the already existing libraries for communicating with our customer's equipment (and also our customer's customers' equipment).

A. User Interface

The graphical user interface displays all relevant information in an intuitive, consistent and visually appealing way (we are aware that the terms used in this description are vague and subject to different interpretations). It removes the demand for any other user interface, and in this way simplifies the interactions to be made with the equipment. The user interface is based on touch screen interactions.

Different users are going to utilize the interface in different ways. Patients and relatives, who will only have access to and use a small amount of the capabilities of the system; nurses, who will unlock additional features of the interface and set up rules and monitor the well-being of the patient; and service technicians who will unlock features for setting up and monitoring the general state of the medical equipment.

The user interface is divided into different modes or screens that have been designed to fit the workflows of that particular user group. The screens are visually clear and concise, making interactions easy in the environments where the user interface is placed.

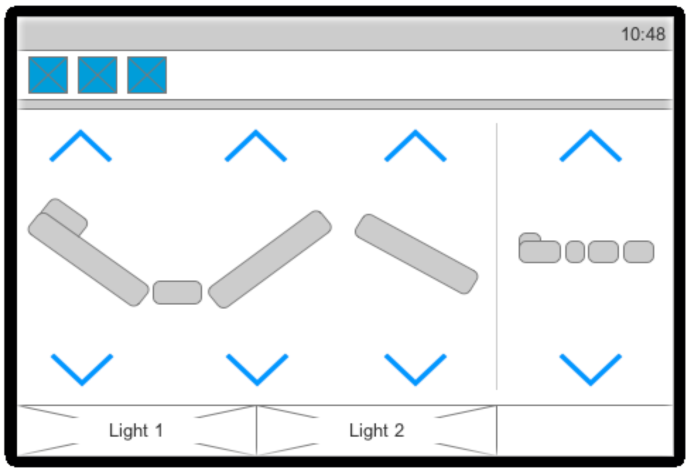


Fig. 2. Screen from the user interface (anonymized and simplified).

The user interface will adjust itself to the capabilities of the equipment. If a particular accessory is present, controls for interacting with it will automatically be available in the user interface.

The product will be used in a global setting and therefore the user interface should be international. This is particularly demanding for the textual design of the interface. In addition, it implies that regional settings, such as time, date and units should be adjustable.

Fig. 2. shows an example of a screen from the user interface, anonymized and simplified for the purpose of this paper.

B. Product Environment

In addition to displaying the graphical user interface the product also communicates with external equipment, such as motors, lights and sensors via a proprietary protocol, and handling interactions with physical buttons, light sensors, RFID chip, buzzers, etc.

As stated, the product is a graphical front end to an entire system of hospital equipment, actuators, motors, sensors and controls as illustrated in Fig. 3. , which shows our product in its technical environment.

Each accessory is a product and the controller, which controls the equipment movements, is also itself a product. The accessories and the movement controller can have a number of sensors attached and are responsible for different parts of the overall system functionality.

Examples of the accessories could be wet sensors, for detecting when a bed is wet, indicating that a change of sheets is required, or a presence sensor which monitors if a patient enters or exits the bed, being of great importance when dealing with patients suffering from, e.g. dementia.

Some accessories can be configured to react to each other. As example, one accessory can detect when a patient is present in the bed and another accessory provides light under the bed, and can be configured to turn on the light when the patient exits the bed. For this to work, all parts of the system must agree on a communication protocol to exchange state and commands between products in the system.

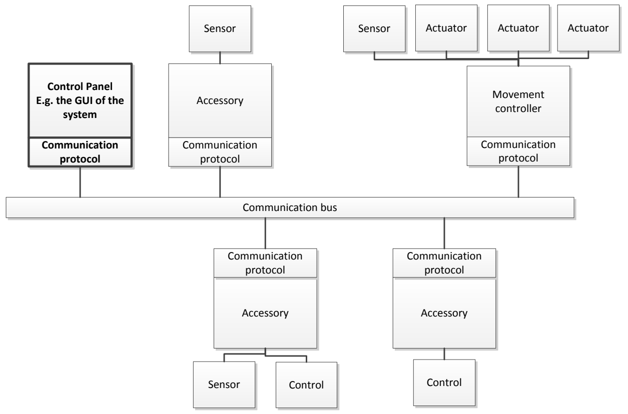


Fig. 3. Our product in its technical environment.

IV. Requirements Specification for the GUI

In this section, we describe the different requirements artifacts used in the development project. The specifications are concerned with product-level requirements, design-level requirements, domain-level requirements and other requirements, as we will discuss below.

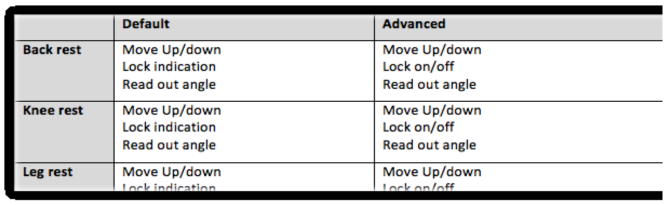
The user experience specialist from Mjølner wrote the specification of product-level requirements (as we have seen previously, the first version of the product-level requirements was conveyed by the customer to us orally). It was based on the prototype and input obtained from workshops with our customer. It was a document which, in a structured way, listed the functionality and status information available in different parts of the GUI. It also described three different operational modes, based on the user profile, e.g. only a service technician could change the language and only a nurse or caretaker could disable specific movements. The structure of the specification supported description of functionalities and dependencies between functionalities in the different modes.

The main purpose with this specification was to be used as a checklist, as a supplement to the prototype, and to give the customer and us a common understanding of what the scope was and how the specific functionality should work.

An overall product-level requirement to the GUI was to reflect the state of the whole system, i.e. all the accessories and the equipment controller. The GUI should also be used to configure and control the accessories and the equipment controller. Fig. 4. shows an extract of the product-level requirements specification.

The design-level requirements were represented using interaction design and graphical design. In the process of going from product-level to design-level, new details about the requirements naturally appeared and our understanding of the requirements was improved.

Domain-level requirements were only discussed informally at a meeting with some documentation in PowerPoint slides.



	Default	Advanced
Back rest	Move Up/down Lock indication Read out angle	Move Up/down Lock on/off Read out angle
Knee rest	Move Up/down Lock indication Read out angle	Move Up/down Lock on/off Read out angle
Leg rest	Move Up/down Lock indication	Move Up/down Lock on/off

Fig. 4. Extract of the requirements specification.

Moreover, there was a number of requirements, which did not relate directly to the GUI. They included communication protocols, interface specifications, the hardware platform and use of code libraries from our customer. Those requirements were handled as "technical issues", which were dealt with in an ad hoc way without structured, written specifications, but via discussions at meetings and through phone and email conversations.

V. Requirements Problems

Above, we have given some background about the project, the product and the requirements specification. We now continue by discussing a number of problems related to requirements that we have encountered. For each problem, we describe what we actually did and we consider alternative actions, we could have taken, including a discussion of pros and cons.

A root cause to the problems we will discuss below is that we did not manage to establish an effective "communication infrastructure" early in the project. Mjølner's main communication link was with the customer's project manager. Initially, it was our impression that he was project manager for the entire project, including the requirements and design part. However, it turned out that in practice he was almost only concerned with the development part. When we had questions related to requirements and design, we were referred to the sales department. This meant that many requirements related issues were dealt with ad hoc, not systematically and not always with as high a priority as we desired.

A. Goal-Level Requirements were not Agreed and Stated Explicitly

At the kick-off meeting for the development project in March 2014, together with the customer, we created a first version of a risk list, which included the following item: The customer's sales and development departments have different and potentially conflicting expectations. The risk occurred and with severe consequences.

We did experience that the customer's sales and development departments indeed had different and conflicting expectations. As mentioned previously, we had the impression that it was a goal-level requirement for the sales department that the new product should help to gain market share on the global market, including in China.

However, we never ensured that the goal-level requirements were agreed upon and stated explicitly.

In retrospect, we should have insisted on that. An obvious advantage of this would have been that it would have helped us in our discussions with the project manager about scope changes. The main goal-level requirement, perhaps more properly named *constraint*, from the development department was that development of the new product should not be too expensive. These two goal-level requirements were in conflict (which is not unusual, but the situation was more severe here than in many other projects we have been involved with).

Moreover, an explicit statement of the goal-level requirements could have catalyzed valuable discussions between and inside our customer's sales and development departments and ultimately a common understanding of the project's goals (and constraints) and a better basis for making the necessary trade-offs.

B. Domain-Level Requirements had too Little Attention

Only to a low degree did we carry out activities that ensured correspondence between goal-level requirements and domain-level requirements.

The first problem related to this is that, as we saw above, the goal-level requirements were not explicitly stated, which gives alignment an unclear starting point. However, if we assume that the goal-level requirements indeed were to strengthen our customer's global market position, then it is a weakness that we only conducted field studies and elicited domain-level requirements in the customer's local market, not in a number of markets, ideally worldwide.

Early in the project, it was discussed whether we should have participated in meetings with our customer's customers to elicit requirements and wishes from the market. If this activity had been carried out, it could have resulted in a better insight or input which might have helped to find and understand the domain-level requirements. This could have provided useful input to our concrete work with design-level requirements.

In many projects, we create scenarios and personas. We did not do it in this project, but it might have been helpful, because they could have served as a common reference and an efficient means of communication. These artifacts might have been very useful in aligning requirements between the sales and development department.

More user tests during the project might have been helpful to validate that the development project was progressing in the right direction. Moreover, we should have tested on a larger and more diverse set of users, including international users. Perhaps we should have insisted harder - even though it would have increased our time consumption here and now, but the advantages we would have gained are likely to have been worth the extra investment.

C. Product-Level Requirements for Several Components were Missing or Incomplete

As we described in section III.B, our GUI controls a system, which is built of multiple products. From the end users perspective, the product is the entire system of hospital equipment, actuators, motors, sensors and controls.

To discuss requirements in this context, we divide the definition of product-level requirements into two sub-categories, *system-level* requirements and *component-level* requirements, where system-level requirements pertain to the entire system and component-level requirements pertain to the individual products, of which the system is built. In general, requirements to the components must be specified in a way, which allows the system-level requirements to be fulfilled when the components are combined to form the system.

For the whole system (see Fig. 3.) comprised of a number of individual products, the product-level requirements of the GUI should be aligned with the product-level requirements of the individual products in order to end up with a coherent system. In other words, the GUI product-level requirements should be aligned with system-level requirements.

An example of this could be a heart rate monitor, which activates a nurse call if the heartrate drops below a certain threshold. If the interaction design shows that the monitor can be configured to deactivate the nurse call if the heart rate returns to normal, but the heart rate monitor product does not have this capability, there is a misalignment between the system-level requirements and the component-level requirements for the heart rate monitor.

We tried to address the alignment issue in workshops with our customer, but our customer thought that it was out of scope for our work and should be handled internally by them. In effect, this meant that the capabilities of the individual products indirectly influenced our GUI requirements, but the products were to be developed by another department at our customer, and we did not have any communication channels to that department. We experienced that product-level requirements for several components were either missing or incomplete.

D. Product- and Design-Level Requirements were not Communicated Sufficiently Clearly

Communication of product- and design-level requirements, across our customer's organisation was a problem.

We had an early meeting with our customer, where we discussed the product-level requirements. The head of software development and a person from the sales department were present. With this attendance, it should have been possible to emphasize the importance of requirements alignment across different departments and to agree how this alignment should be done continuously throughout the project.

Unfortunately, we did not manage to achieve this. Product-level requirements were not communicated properly and actively used in our customer's organization. It is our impression that our customer do not have much experience with working with requirements in this way – and we did not manage to make things work.

Our customer's sales department are not used to writing and maintaining a specification of product-level requirements as the one we presented in Section IV. They see it as viable to quickly develop something with the "right" set of features, without an extensive prior analysis; and they are ready to change it afterwards if necessary. In contrast, in the development department, they are aware that specifications often are important; and in some cases absolutely necessary, e.g. in relation to getting certifications for certain products. A problem, though, is that our main audience for our product-level requirements specification was the sales department.

Thus, we did not always succeed in communicating the product-level requirements to our customer.

We had different experiences regarding design-level requirements. Here, our customer were interested and ready to provide feedback. We had continuous iterations with the sales department who frequently provided feedback. In particular, the feedback became very concrete when they saw the graphical design and when they experienced the implementation.

The feedback implied that we went back through levels of the requirements process. We first updated the graphical design to achieve approval from our customer, then we updated the interaction design according to the graphical design and the implementation, and at last we updated the product-level requirements specification, although the interaction design and the product-level requirements specification were no longer actively used in the process.

In general, we believe that we should have worked with more specific representations earlier. It might well have been more effective. If this is correct, we would have used less time on rework during implementation. Often, we started implementation of various features before the design was finally approved.

It is our customer's sales department, which is responsible for approval, based on their subjective decisions and immediate response. Sometimes, they were very busy and we did not make our case with sufficient conviction, when we tried to argue for different decisions. A simple correction in the graphical design could often imply that the implementations task would grow. This gave us some challenges and it was difficult for the development department at our customer to understand why the estimate was increasing again. The development department often did not know about the changes before being informed by us.

We should have insisted that our customer's development department had validated the product- and design-level requirements, and insisted on making a walk-through of in particular the design-level requirement specification with both the sales and the development people, preferably gathered in the same room.

E. The Technical Environment was Unstable

All existing products and the previously developed prototype used a simple communication protocol and in our project proposal we assumed that the GUI should also use this protocol. For description purpose, we will call this *protocol 1*. This was a simple protocol, but it needed a lot of configuration of each component to make the components work as a system.

At the full product development project kick-off meeting in March, we were told by our customer, that they planned to design and implement a new communication protocol. This we will call *protocol 2*. This new protocol would be much more advanced than *protocol 1* and provide automatic discovery and configuration of the components in a system. The new requirement for our GUI was that it should work in a system, where some products used the old protocol and some products used the new protocol.

At the meeting, we were also informed that some of the accessories to be used in the system, i.e. the components of the system, were planned to be developed in parallel with our GUI and that they would use the new communication protocol. The detailed functionality of these accessories were yet to be specified. Some of the existing accessories, using *protocol 1*, would also be used in the system.

Only one customer employee was assigned to define *protocol 2* and he was only part-time. The development of new accessories was the responsibility of another department at our customer. The schedule was unknown to us and we did not have direct communication with this department.

We never identified a person from the customer, who was responsible for defining the component-level requirements, which were necessary to fulfil the system-level requirements. As a result of this, the component-level requirements were either missing or incomplete. This implied that it was very difficult for the person, who should define the communication protocol, to make progress, because of a lack of overview of what should be communicated between components.

We had a preliminary version of the protocol specification, which we used as a basis when we started our implementation. We agreed on delivery dates for updated versions of the protocol and accessory specifications, but they were delayed. Before the summer vacation, we had a meeting with our customer, where we expected to be given an updated version of the specifications. But instead we were informed, that the development of *protocol* 2 would be replaced by development of another new protocol. We will call this *protocol 3*. This protocol would use some of the same principles as *protocol 2* but it would no longer be able to exist in parallel with *protocol 1*. The new requirement for our GUI was to use *protocol 3* and our customer would provide specifications and an implementation of the protocol and new accessories, which used the new protocol.

During the next months, we requested specifications and implementations, but they were delayed. Mjølner's software architect had a one day work-meeting with the protocol responsible from the customer, where they discussed some of the principles in the new protocol.

In August, we were told by our customer, that they would not be able to develop new accessories based on *protocol* 3 before the November exhibition. Instead, they planned to build a system based on *protocol 1*, but using a stricter and predefined configuration. This was labelled *protocol 1.1*.

The requirement for our GUI was now to be implemented as if it was using *protocol 3* and to have an adapter layer, which converted between *protocol 3* and *protocol 1.1*.

In September, we still did not have any specifications of either *protocol 3*, *protocol 1.1* or the new accessories. The architect from Mjølner suggested that our GUI should use *protocol 1.1* directly and thus reduce the remaining specifications to only what our customer had plans for implementing. Our customer agreed that this was the best solution.

Later in September, we had a meeting with our customer, where we discussed the *protocol 1.1* specification. The person, who should specify the protocol, had not been provided with an overview of neither the system-level requirements or the component-level requirements, i.e. he had not been told how the system was supposed to work and had not been informed what each accessory should be capable of.

We discussed the principles of the protocol and in the following week, the architect from Mjølner created a suggestion for a protocol specification based on our GUI specification. The specification included component specific details and thus imposed component-level requirements, which were written in a way so that the system-level requirements would be fulfilled. This specification was used as basis for further work by our customer and in late September, the first draft of the *protocol 1.1* specification was delivered to Mjølner. We implemented the communication layer of the GUI according to this specification and after a few iterations, we had a working system and a final version of the specification.

From the previous description, it can be seen that the environment in which our GUI was to exist was not stable.

At the kick-off meeting, it was clear that not all the system components existed and that the protocol to be used was unknown and under development. This was identified as a major risk by our customer's head of software development. This was clearly a deviation of the assumptions in our original proposal, but regardless of that, we agreed to use this new protocol and to depend on accessories, which were under development.

In retrospect, we might have insisted that the technical environment was kept more stable or we should have communicated more clearly about the consequences of the proposed changes. Alternatively, we could have chosen not to implement anything in the communication parts of the GUI until the final specifications were provided to us. This would have forced us to postpone work and to allocate people to other projects, which would reduce the chance of having a working product ready for the exhibition in November. There would have been a large risk of not being able to meet the deadline, but there would have been more direct impact on the customer, which might have increased the chance of necessary action from their side.

VI. Requirements Problems Contributed to Increasing Estimates

The problems we have described above all emerged during the development of the full product; the prototyping and field studies were successful in their own right. The problems described in sections V.C, V.D and V.E directly implied increased estimates and thus decreased customer satisfaction. The problems described in sections V.A and V.B contributed more indirectly. If we had had explicitly stated goal-level requirements and worked more thoroughly with domain-level requirements we had been in a better position to make and justify our choices during the project including the choices affecting the increases in estimates.

Often we did not have the necessary knowledge to deliver precise estimates. Sometimes the circumstances required us to deliver estimates anyway; often we did not succeed in communicating the high uncertainty in our estimates clearly. Initially, the customer's project manager understood and accepted that an extended scope for the product implied increased time estimates for the development. However, at a certain point, this acceptance became more difficult to get, and it was much harder to get acceptance of increased estimates due to the other factors discussed in Section V.

We accepted this state of dissatisfaction on project level for too long. We merely described the reasons to increased estimates in our weekly status reports, and we indeed did increase our estimates quite frequently. We did not have the instruments to tackle the problems properly on project level – and there was no organization in place above the project, e.g. there was not a steering committee, empowered to resolve the issues that caused the problems.

The project might have had a different outcome, if we had insisted on attention and action from higher levels in our respective organizational hierarchies. As an example, perhaps we should have stopped the project, when we could see that the instability in the technical environment was too severe. Also, we might have insisted that our original agreement was to build a GUI based on existing technology and that we should create a new contract and new estimates due to this change. Had we done that, the contract change would probably have reached a higher level in our customer's organisation than the change of scope and the increased estimates of the existing contract did.

An advantage of this would obviously have been that our time consumption, and thus our invoice to the customer, would have been significantly lower. A disadvantage might have been that we would not have tried as much as we could to help the customer with making a good product ready for the important international exhibition in November.

Another less drastic alternative could have been to use a more formal change management procedure, which might have contributed to keeping and communicating a better overview of the current state of the project.

VII. Conclusions and Related Work

There are a number of papers, which describe successful requirements engineering and software development, also under difficult project circumstances including changing requirements, e.g. [2] and [7].

From our company, we have over the last few years published, e.g. [1], [4] and [6]. With the paper at hand here, we supplement these "success stories" by describing a project, which did not have the desired outcome. Even though this is unfortunate, it is not unusual. It is well known that a large fraction of software projects fail, e.g. the product is never delivered, a bad product is delivered or there are significant overrun in deadline and budget. In our case, the latter happened.

The problems we discussed in Section V, all first emerged in the development of the full product. As we saw, the prototyping and field studies were done to our customer's full satisfaction. However, the problems seem to have their origin in the very, overall process, we followed during the project. As described in Section II, the sequence of activities carried out can be described like this: (1) prototyping, (2) field studies (3) development of a full product. This is not the "textbook perfect" sequence. It is more common to do this sequence: (a) field studies (b) prototypes (incl. validation), (c) development of full product; see, e.g. [8] or [10].

The actual sequence we carried out was determined by commercial circumstances. Our first contact with the customer in the context of the project described in this paper was about development of a prototype, not about conducting field studies etc. More generally, often prototypes or something similar will be a starting point. It is easier to sell than field studies etc., because in the latter case, the customer spend money on Mjølner building up domain knowledge, not on Mjølner developing something technological that the customer cannot do themselves. However, in summary, it might be argued that we carried out main activities in the wrong sequence, with the effect that our very foundation for requirements was too fragile.

Another perspective on this can be seen by considering our work with the different levels of requirements. Our sequence of treatment of requirements levels were: (1) product-level – delivered orally to us as input to the prototype, (2) design-level – made by us via the prototype, (3) domain-level – discovered by us through field studies etc. Somehow on the side were the goal-level requirements, which were more implicit than the other requirements-levels. Again, the "textbook perfect" sequence of treatment would be: (a) goal-level, (b) domain-level, (c) product-level, (d) design-level; see, e.g. [5].

We believe that the non-standard sequences described above have contributed to the problems we have encountered in this project. Our foundation has not been as solid as it should, and this has added to misunderstandings and mis-alignments in our cooperation with the stakeholders from the customer. The problems emerged when our set of customer stakeholders increased from one person during the prototyping and field studies activities to several persons and departments in the full product development project.

In general, when we develop software, we are concerned with making an argument like "(A and S) implies R", where A is assumptions about the environment, S is a specification of our software (product or design-level requirements) and R is domain-level requirements. Many authors have described this, see, e.g. Jackson [3] and Wieringa [11]. Wieringa refers to the implication above as the "software engineering argument". A main problem in the project under discussion in this paper is that "A" (accessories, communication protocols) was very unstable, where in other projects, "A" is relatively stable, and "R" for the most part is found before "S"; this was not the case in our project.

To sum up, this paper is a Problem Statement, and as such of a speculative nature. If we had acted differently in the project, we of course do not know what would have happened. We do believe that the requirements work under similar circumstances should be carried out more systematically and with higher priority. It is likely that it would have reduced our problems. In addition, a stronger project organization (steering committee), stronger project management and better stakeholder handling, would probably have alleviated our problems even more.

ACKNOWLEDGMENT

We thank our colleagues, who have commented and suggested improvements to this paper.

REFERENCES

[1] L. Bruun, M. B. Hansen, J. B. Iversen, J. B. Jørgensen, B. Knudsen, ”Handling design-level requirements across distributed teams: developing a new feature for 12 Danish mobile banking apps”, 22nd IEEE International Requirements Engineering Conference (RE14), Karlskrona, Sweden, 2014

[2] D. Hauksdottir, A. Vermehren, J. Savolainen, “Requirements reuse at Danfoss”, 20th IEEE International Requirements Engineering Conference (RE12), Chicago, Illinois, 2012

[3] M. Jackson, Problem Frames – Analyzing and Structuring Software Development Problems, Addison Wesley, 2001

[4] J. B. Jørgensen, K. Nørskov, N. M. Rubin, ”Requirements engineering and stakeholder management in the development of a consumer product for a large industrial customer”, 19th IEEE International Requirements Engineering Conference (RE11), Trento, Italy, 2011

[5] S. Lauesen, Software Requirements - Styles and Techniques, Addison Wesley, 2004.

[6] M. S. Loft, S. S Nielsen, K. Nørskov, J. B. Jørgensen, ”Interplay between requirements, software architecture and hardware constraints in the development of a home control user interface”, Twin Peaks workshop at 20th IEEE International Requirements Engineering Conference (RE12), Chicago, Illinois, 2012

[7] A. J. Nolan, S. Abrahão, P. Clements, A. Pickhard, “Managing Requirements Uncertainty in Engine Control Systems Development”, 19th IEEE International Requirements Engineering Conference (RE11), Trento, Italy, 2011

[8] S. Robertson, J. Robertson, Mastering the Requirements Process, Second Edition, Addison Wesley, 2006

[9] H. Sharp, Y. Rogers, J. Preece, Interaction Design, John Wiley & Sons, 2007

[10] I. Sommerville, Software Engineering, 8th Edition, Addison Wesley, 2007

[11] R. J. Wieringa, Design Methods for Reactive Systems – Yourdon, Statemate, and the UML, Morgan Kaufmann, 2003