Janna Ylönen

USER REQUIREMENTS ELICITATION WITH INTERVIEWS AND PROTOCOL ANALYSIS METHODS

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2022

ABSTRACT

Ylönen, Janna
User requirements elicitation with interviews and protocol analysis methods
Jyväskylä: University of Jyväskylä, 2022, 62 pp.
Information Systems Science, Master's Thesis
Supervisor: Kujala, Tuomo

The purpose of this study is to describe the implementation of two user requirements elicitation methods, Interviews and Protocol Analysis, analyse the gathered material and to derive problems and user requirements from it to be used in a system upgrade, and to finally compare the results from the methods.

A total of 8 sessions were conducted as online meetings that were recorded. All of the participants were previously familiar with the system and each in session, both of the methods were executed. The recorded material was analysed so that issues with the system or the surrounding processes were identified from the users' input. The issues were considered based on what could be the underlying problem causing them or what problems they could lead to. Finally, user requirements were derived from the problems, suggesting what would solve the problems. The problems and user requirements were then divided into categories and their frequencies were counted.

The participants were more talkative in the conducted Interviews even though they were instructed and encouraged for thinking aloud while performing the Protocol Analysis tasks. The Interviews yielded more user requirements in total, and for an example, to categories such as Processes where we focused on finding out the processes surrounding and related to the system. With Protocol Analysis we gathered more user requirements related to the system's usability, as the participants were more likely to remember and point out problems with the system while interacting with it. Problems and bugs in the system could also be recognised by only observing the participants' actions in the system which would not have come up in the interview. This way the two methods were seen to complement each other.

The number of gathered user requirements was extensive even with only 8 participants. As it was one reason for selecting the Interviews and Protocol Analysis methods, this study also demonstrated that the two methods do not require a large sample size due to the richness of the collected data, especially when combined.

Keywords: User requirements elicitation, Elicitation methods, Requirements engineering, Interviews, Protocol Analysis, Selecting, Comparison.

TIIVISTELMÄ

Ylönen, Janna
Käyttäjävaatimusten kerääminen haastattelu- ja ääneen ajattelu -menetelmillä
Jyväskylä: Jyväskylän yliopisto, 2022, 62 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja: Kujala, Tuomo

Tämän tutkimuksen tarkoituksena on kuvata kahden käyttäjävaatimusten keräysmenetelmän, haastattelujen ja ääneen ajattelun toteutus, kerätyn materiaalin analysointi, siitä käyttäjävaatimusten johtaminen käytettäväksi järjestelmän jatkokehityksessä sekä lopulta vertailla menetelmien tuloksia.

Kaikkiaan 8 istuntoa järjestettiin verkkokokouksina, jotka nauhoitettiin. Järjestelmä oli kaikille osallistujille ennestään tuttu ja jokaisessa istunnossa toteutettiin molemmat menetelmät. Tallennettu materiaali analysoitiin siten, että käyttäjien puheesta ja toimista järjestelmässä pyrittiin tunnistamaan järjestelmään liittyviä haasteita. Haasteita tarkasteltiin sen perusteella mikä voisi olla niiden taustalla oleva ongelma tai mihin ongelmiin ne voisivat johtaa. Tämän jälkeen ongelmista johdettiin ehdotuksia käyttäjävaatimuksiksi, jotka voisivat toimia ratkaisuna löydettyihin ongelmakohtiin. Lopuksi sekä ongelmat että käyttäjävaatimukset jaettiin kategorioihin ja laskettiin.

Osallistujat olivat puheliaampia haastatteluissa, vaikka heitä ohjeistettiin ja kannustettiin kertomaan ajatuksistaan ääneen ajattelu -menetelmän tehtäviä suorittaessaan. Kokonaismäärältään enemmän käyttäjävaatimuksia saatiin kerättyä haastatteluista esimerkiksi Prosessit-kategoriaan, johon luokiteltiin vaatimuksia, jotka liittyvät järjestelmään ja sitä ympäröiviin prosesseihin. Ääneen ajattelun avulla löydettiin enemmän järjestelmän käytettävyyteen liittyviä käyttäjävaatimuksia, osallistujien muistaessa ja kertoessa järjestelmään liittyvistä ongelmista paremmin heidän samalla käyttäessään järjestelmää. Myös vain tarkkailemalla osallistujien toimia voitiin huomata ongelmia ja vikoja järjestelmässä, joita ei olisi tullut esille haastatteluissa. Näin näiden kahden menetelmän voitiin todeta täydentävän toisiaan.

Kerättyjen käyttäjävaatimusten määrä oli suuri, vaikka siihen osallistui vain 8 henkilöä. Yksi syy haastattelu ja ääneen ajattelu -menetelmien valinnalle oli, että ne eivät vaadi suurta otoskokoa kerättyjen tietojen monipuolisuuden vuoksi. Tämä voitiin myös tässä tutkimuksessa todeta, varsinkin menetelmät näin yhdistettäessä.

Asiasanat: Käyttäjävaatimusten kerääminen, menetelmät, vaatimusmäärittely, haastattelut, ääneen ajattelu, valitseminen, vertailu.

FIGURES

TABLES

CONTENTS

1 INTRODUCTION

User requirements elicitation is an essential phase in product development. In this phase, all the stakeholders using the system and the processes involved are identified and understood. User requirement elicitation, or sometimes called requirement collecting techniques are methods used to determine the needs of users so that the systems can be built with a high probability of satisfying their needs. In other words, user requirement elicitation ensures that the product that is being built is of the right type.

1.1 Motives for the study

The success or failure of a system development effort depends heavily on the quality of the requirements (Jones, 1996). The quality of the requirements is greatly influenced by methods used in requirements elicitation because elicitation is all about learning the needs of users and communicating those needs to system builders (Hickey & Davis, 2003).

Before developing any system, we must understand what the system is supposed to do and how its use can support the goals of the individuals or business that will pay for that system (Sommerville, 2005). We must find out the users' needs in order to increase the quality of a system. No requirements elicitation technique has the capability of finding all of the user requirements. To gain a more effective elicitation, using a variety of techniques will help in gathering more complete and correct requirements (Yousuf & Asger, 2015).

There are plenty of literature describing requirements elicitation methods and providing instructions for their use. Some research can be found where different user requirements elicitation methods are compared (Carrizo, Dieste, & Juristo, 2014; Goguen & Linde, 1993; Maiden & Rugg, 1996; Yousuf & Asger, 2015; Zowghi & Coulin, 2005). That research mainly focuses on comparing the implementation of the methods, and not on comparing the results, that would be for example, the number or relevance of the elicited user requirements.

1.2 Research problem

The research problems in this study are the following:

- How did the outcomes of the implemented user requirement elicitation methods differ?
- Which of the implemented user requirement elicitation methods proved to be better considering the returns management system upgrade?
- Which of the implemented user requirement elicitation methods yielded more higher priority user requirements for the system?

1.3 Research method and objectives

In this study we implement two user requirement elicitation methods: Interviews and Protocol Analysis. From the data gathered with these methods, we search, and list found problems related to the system's processes, usability, reporting, integrations, and information security. The problems are then further formed into user requirements for the system.

This study demonstrates one way to compare the results of user requirement elicitation methods and how to deduce based on the results (found problems, and derived user requirements) which one performed better. This thesis is a comparative study that acts as a preliminary analysis for a system upgrade.

1.4 Structure of the thesis

This thesis is organised as follows. Chapter 1 presents the background and motives for the study, research problem, method and objectives and the structure of this thesis. Chapter 2 is a literature review that describes the role of requirements engineering in system development, the requirements engineering process, and then focuses on eliciting user requirements, how the elicited requirements can be then categorised, prioritised and further validated and finally summarised in a requirements specification document.

In Chapter 3 we present a way to categorise different user requirement elicitation methods. The elicitation methods are then presented in a table where we list the pros and cons found from literature for each of the methods. Based on this comparison we selected the Interviews and Protocol Analysis methods for implementation for this study. These methods are then further described.
Chapter 4 firstly summarises the research questions. Then we focus on describing the product Return system that we are gathering the user requirements for – what the main problems are with the current system and the reasons behind the need for the system upgrade. We describe the roles the users of the Return system

work in and illustrate the main processes surrounding the system step-by-step: what the different users of the Return system do to get their part of the job done from reporting faulty devices to the system, to the Company sending replacing devices for the Customer. We describe the procedure how the methods were implemented and how the gathered data was processed and analysed.

In Chapter 5, we firstly describe the participants' background information. Then we present the results of the two requirements elicitation methods, Interviews and Protocol Analysis. The counted frequencies of problems, problems per participant and requirements are shown in tables per category. Finally, we describe what types of problems and requirements were gathered with the methods to each category.

In Chapter 6 we answer to the research questions, present and discuss of the findings of the study, and reflect the findings against the literature review in Chapters 2 and 3. In Chapter 7 we will assess the reliability and the validity of the results as well as the usability and the significance of the results for the future, follow-up research and conclude this thesis by considering what the next steps to continue this work could be.

2 USER REQUIREMENTS ENGINEERING

In this chapter we describe the user requirements engineering process, its role and purpose in software development as a whole. We describe what the common process steps are, and as this thesis focuses on eliciting and categorising user requirements, those activities are described in more detail. Finally, we describe what the final outcome of the process, the requirements specification document is, and what it consists of.

2.1 Role of requirements engineering in software development

System requirements are specifications of the service the system should provide, the constraints on the system and background information which is necessary to develop the system. *Requirements engineering* is the systematic process of eliciting, understanding, analysing and documenting these requirements. The term engineering is used to indicate that this is a practical, systematic process that ensures that the system requirements are complete, consistent and relevant where tradeoffs have to be made to find the best solution. (Kotonya & Sommerville, 1998.)

According to Kotonya and Sommerville (1998) there can be a number consequences for when the user requirements for a system are incorrect:

- Delays in the delivery of the system and increased costs
- The systems unreliability, with recurring error situations and crashes that interfere with daily use
- Customer and user dissatisfaction with the implemented system, they dislike it and may discontinue using it
- Costs of maintaining and updating the system rise to be very high

A great deal of research has verified that devoting systematic effort to requirements engineering can greatly reduce the amount of rework needed later in the life of a software product, and that it can cost-effectively improve various qualities of the system (Laplante, 2009).

2.2 Requirements engineering process

Requirements engineering process is a structured set of activities which are done to establish, validate and maintain a systems requirements document (Kotonya & Sommerville, 1998). In addition, requirements engineering process is where the visions about a system are established in a context (Pohl, 1994).

There are many different aspects and viewpoints to requirements engineering and different books divide the activities in different ways. Similarly, there is no single process that is right for all companies. The process used for requirements engineering vary widely depending on the types of systems a company develops, company culture, and the level of experience and ability of the people involved in requirements engineering.

However, there are number of generic activities common to all processes. One common way to describe the activities is described by Sommerville & Sawyer (1997). Their model of requirements engineering consists of activities that cover discovering, analysing, documenting, and maintaining a set of requirements for a system.

Sommerville (2005) presents the activities of requirements engineering as a cycle, where individual activities repeat as the software requirements are derived, and the iteration continues during system implementation and operation. The activities repeat in a cycle and in addition, requirements documentation and requirements management are done always on the side as their own process.

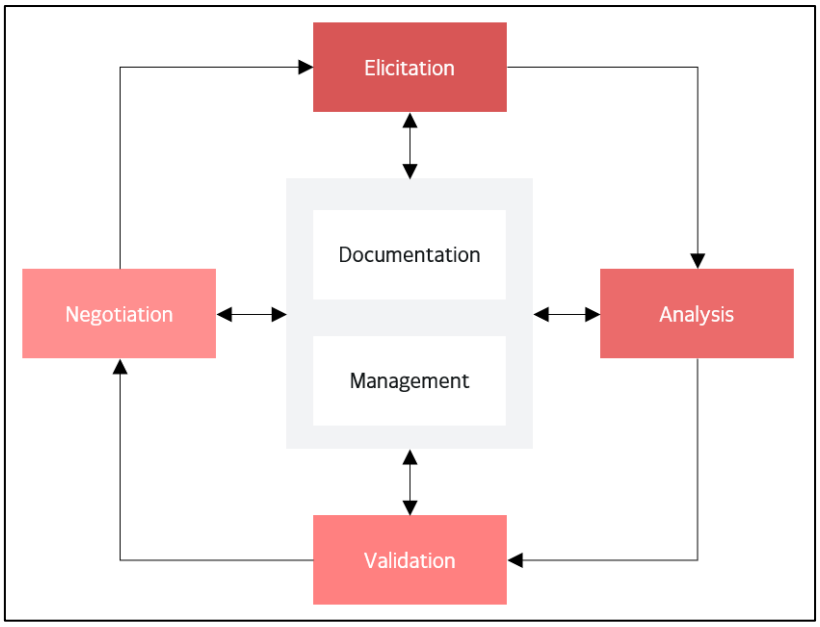

FIGURE 1 Requirements engineering process (Sommerville, 2005)

The activities can be described shortly as follows. The activities include firstly requirements *elicitation*, which means identifying the sources of information about the system and discovering the requirements from these. Then the

requirements' overlaps, and conflicts are *analysed*. In user requirement *validation* we go back to the system stakeholders to check that the requirements are what they really need. *Negotiation* activity aims to reconcile conflicting views and to generate a consistent set of requirements as inevitably the stakeholders' views will differ, and the proposed requirements might conflict. In *documentation* the requirements are written down in a way that stakeholders and software developers can understand. Requirements *management* controls the requirement changes that will arise. Requirements for a system always change to reflect the needs of stakeholders, changes in the installation environment, in the business or to laws or regulations etc. (Sommerville, 2005.)

2.3 Requirements elicitation

Requirements elicitation or the discovery of user requirements, comprises an early and critical but highly error-prone stage in system development (Coughlan & Macredie, 2002). Goguen and Jirotka (1994) note that the elicitation of user requirements is perhaps the activity that is most often regarded as the first step in the requirements engineering process.

One typical definition for a *user requirement* can be found from the IEEE Standard Glossary of Software Engineering Terminology: A requirement is a condition or capability needed by a user to solve a problem or achieve an objective (IEEE, 1990). Robertson & Robertson (1999) state that a requirement is something the system must do or a feature that the system must have.

Elicitation means "to call forth or draw out (something, such as information or a response)" as defined by the Merriam-Webster (2022) dictionary. So, the practitioners aim to draw out the requirements from the stakeholders. According to Goguen and Jirotka (1994), the term "elicitation" is preferred to "capture", to avoid the suggestion that requirements are out there to be collected simply by asking the right questions.

Requirements elicitation is how the needs of customers and users are found out, so that systems can be built with a high probability of satisfying those needs (Hickey and Davis, 2003). One of the most important goals of elicitation is to find out what problem needs to be solved, and hence identify system boundaries (Nuseibeh & Eastbrook, 2000). Information gathered during requirements elicitation often has to be interpreted, analysed, modelled and validated before the requirements engineer can feel confident that a complete enough set of requirements of a system have been collected. (Nuseibeh & Eastbrook, 2000)

2.4 Requirements categorisation and prioritisation

Requirements categorisation is the process of recognising, differentiating, and classifying requirements for some specific purpose and is usually performed

manually (Yue, Briand & Labiche, 2011). One way for dividing requirements into categories is presented by Maguire and Bevan (2002), as they suggest the categories User requirements, Usability requirements and Organisational requirements. *User requirements* are tasks that the system will support and the functions that will be provided to support them. *Usability requirements* include understandability, learnability, supportiveness, flexibility and attractiveness objectives for the system, which should be specified as measurable requirements. *Organisational requirements* deal with supporting the management structure of the organisation and communications within it, as well as group and collaborative working. Legislative requirements may also be categorised as organisational requirements. (Maguire & Bevan, 2002.)

In this study we have derived the categories from what were seen as the main development areas in the system based on the Interviews and Protocol Analysis sessions. The requirements in this study were organised into categories named *Process, General use and usability, Reporting and follow-up, Integrations* and *Information security*. They are further described in Chapter 4.4 *Procedure*.

Requirements prioritisation is important so that development resources can be directed appropriately (Maguire & Bevan, 2002). It requires complex context-specific decision-making and must be performed iteratively in many phases during development work. It can be difficult to pay attention to all the relevant factors that have an effect on priorities and managing the different stakeholder views and customer preferences. (Lehtola, Kauppinen & Kujala, 2004.)

Having systematic requirements prioritisation practices is a challenge as requirements prioritisation requires a great deal of non-trivial decision making. Requirements prioritisation is one of the most crucial and at the same time a difficult task that faces the decision makers. The priority of a requirement is based on many factors such as financial benefit of the requirement for the company, requirement's importance to users, and implementation costs. These factors can be grouped into three main points of views: business, customers, and implementation. These three points of view are introduced in the following figure. (Lehtola, Kauppinen & Kujala, 2004.)

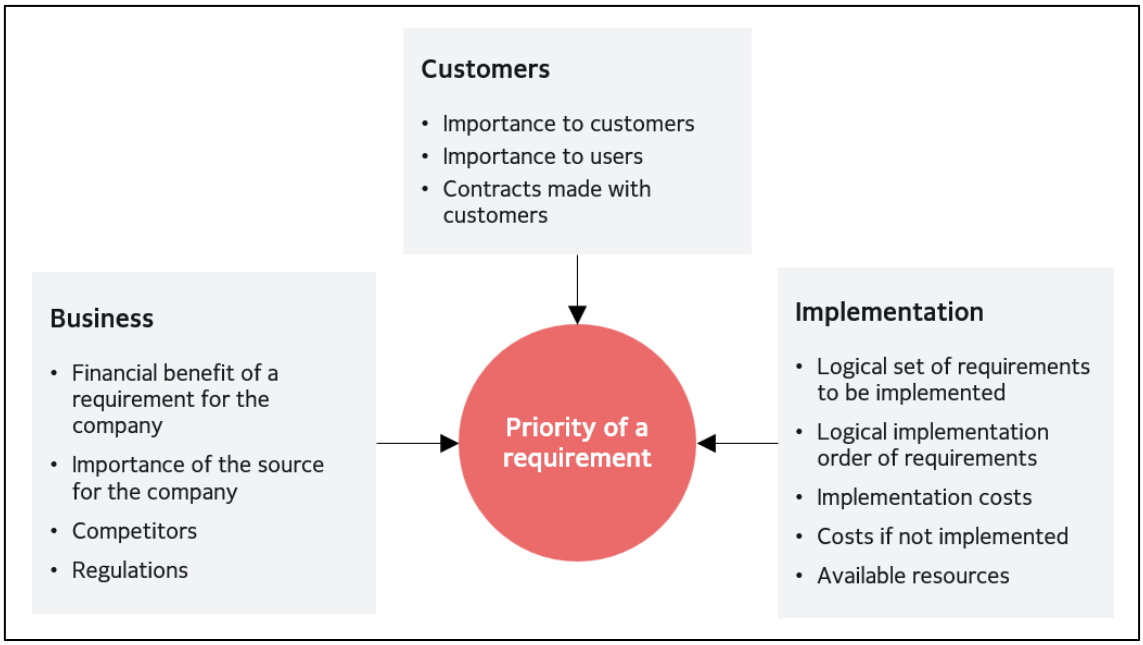

FIGURE 2 Factors that have an effect on requirement's priority (Lehtola, Kauppinen & Kujala, 2004)

2.5 Documentation

The requirements specification document is an official statement of the system requirements that is intended for customers, users and software developers. It is the end product of software requirements engineering. Depending on the company, the document can have different names, for example functional specification, requirements definition or software requirements specification. The requirements specification is the system's specification for the stakeholders and engineers involved in the system development.

According to Kotonya and Sommerville (1998), the requirements specification document should describe the following:

1. The services and functions which the system should provide
2. The constraints under which the system must operate
3. Overall properties of the system, and the limitations on the implementation of the system
4. Definitions of other systems which the system integrates with
5. Information about the application domain of the system, for example how to carry out a particular type of computation
6. Constraints of the process used to develop the system

In addition, the document must be easily altered and serve as source material for system administrators. It must contain an advance view of the future

development of the system and describe the system's responses to unwanted events. (Heninger, 1980.)

A good user requirement is comprehensible, consistent and can only be interpreted in one way (Sommerville, 2000). It is difficult to describe individual requirements. Frequent problems are lack of clarity and mixing and combining requirements with other requirements (Kotonya and Sommerville, 1998).

3 ELICITATION METHOD SELECTION

In this chapter we first present one way to categorise user requirement elicitation methods. The categorisation is then used in the chapter 3.2 *Elicitation method pros and cons*, in Table 1 that was used to help in selecting the user requirements elicitation method for this study. Lastly, we focus on describing the Interviews and Protocol Analysis user requirement elicitation methods as they were the ones selected for implementation.

3.1 Categorisation of methods

One popular way to categorise user requirement elicitation methods is according to Nuseibeh and Easterbrook (2000). They describe the main methods and characteristics of each category as follows.

1. *Traditional methods* include a broad class of generic data gathering techniques. These include the use of questionnaires and surveys, Interviews, and analysis of existing documentation such as organisational charts, process models or standards, and user or other manuals of existing systems.
2. *Group elicitation methods* aim to foster stakeholder agreement and buy-in, while exploiting team dynamics to elicit a richer understanding of needs. They include brainstorming and focus groups, as well as JAD workshops.
3. *Prototyping* has been used for elicitation where there is a great deal of uncertainty about the requirements, or where early feedback from stakeholders is needed (Davis, 1992). Prototyping can also be readily combined with other techniques, for instance by using a prototype to provoke discussion in a group elicitation technique, or as the basis for a questionnaire or think-aloud protocol.
4. *Model-driven methods* provide a specific model of the type of information to be gathered and use this model to drive the elicitation process. These include goal-based methods and scenario-based methods.

5. *Cognitive methods* include a series of techniques originally developed for knowledge acquisition for knowledge-based systems (Shaw & Gaines, 1996). Such techniques include for example protocol analysis, laddering, card sorting and repertory grids.
6. *Contextual methods* emerged in the 1990's as an alternative to both traditional and cognitive techniques (Goguen & Linde, 1993). These include the use of ethnographic techniques such as participant observation. They also include ethnomethodology and conversation analysis, both of which apply fine grained analysis to identify patterns in conversation and interaction (Viller & Sommerville, 1999).

In addition to the above, there are *Model-driven methods* that consist of goal-based methods, such as KAOS and I* and scenario-based methods, such as CREWS that provide a specific model of the type of information to be gathered and use this model to drive the elicitation process. These are supporting methods used for representing and modelling goals or scenarios from separately gathered user requirements. Therefore, these methods were left out of the scope of this study.

3.2 Elicitation method pros and cons

The following table lists different kinds of user requirement elicitation methods in the categories described in the previous chapter. Pros and cons that were found for each requirement elicitation method from different sources are listed in the corresponding columns.

The pros or cons that do not have a marked reference are reflections of the method by the author of this thesis.

TABLE 1 User requirement elicitation method comparison

Cat.	Method name	Pros	Cons
1	Document analysis	Efficient due to that is requires data selection instead of collection. Inexpensive when documents are already available. Documents are unobtrusive, non-reactive and stable. (Bowen, 2009.)	The documents are not produced for research purposes and usually do not provide sufficient detail for it. (Bowen, 2009.) Difficult to implement in requirements engineering when there is little documentation.
1	Existing data analysis	Can provide quick views on the research area. Efficient. Inexpensive. Fast. (Hulley et al., 2013.)	No control over what data have been collected, and how (Hulley et al., 2013).
1	Interviews	More in-depth information. Effective in various situations. (Yousuf & Asger, 2015.)	Structured interviews do not allow generation of new ideas (Zowghi & Coulin, 2005).

Cat.	Method name	Pros	Cons
		Could engage users in the development.	Reach only few users (Yousuf & Asger, 2015). Say-do problem (Goguen and Linde, 1992).
1	Questionnaires	Looks scientific because they use statistical analysis. Useful with a large population. Easily combined with other methods. (Goguen & Linde, 1993.) Useful when the domain is well understood by the stakeholders and the requirements engineer (Laplante, 2009).	Users are prone to state that they will require more features that they will actually use (Laplante, 2009). Questions or the administrative method cannot be changed during the process. (Okwemba, 2019.) A given question has a different meaning for different stakeholders. Shared meaning cannot be established between the stakeholder and the interviewer. (Goguen & Linde, 1993.)
2	Brainstorming	Produces a lot of new ideas and perspectives. Open environment for discussion. (Tiwari et al., 2012.) Easily modifiable, and modifications available (Laplante, 2009).	Large amount of material with no value (Nijstad & De Dreu, 2002) Productivity loss in large groups (Lamm & Trommsdorff, 1973). Evaluation apprehension (Collaros & Anderson, 1969). Freeriding on the efforts of others (Diehl & Stroebe, 1987).
2	Focus groups	Inexpensive. Easy to conduct as users usually like to share their opinion (Simon, 1999). Promotes cooperation, understanding, and teamwork among different stakeholders. Generates complete and valid results to improve work practices. Helps in stakeholders' acceptance and later usefulness of the system. (Farinha & Mira da Silva, 2009.)	Participant recruitment can be expensive and difficult (Krueger & Casey, 2000). Requires an experienced moderator (Farinha & Mira da Silva, 2009).
2	Joint Application Design (JAD)	Improves communication (Wood & Silver, 1989). Brings together stakeholders in different positions (Kettelhut, 1993). Reduces cycle time for systems development. Develops team rapport. Compatible with several development methods. (Duggan & Thachenkary, 2003.)	Dependent on excellent facilitation and control (Duggan & Thachenkary, 2003.) JAD sessions are long, time consuming (Wood & Silver, 1989).
3	Prototyping	Available in an early phase of development process. (Kotonya & Sommerville, 1998.) Permit early evaluation since the prototypes can be tested in various ways (Sears and Jacko, 2009). Users can develop a concrete sense of the system before implementation.	Too high expectations for how ready the system is. Poor quality codes from the prototype may remain in the final system. Lack of mechanisms for requirements traceability.

Cat.	Method name	Pros	Cons
		Allows identification of requirements that may otherwise be impossible. Concretely presents system operations and highlights design decisions. Stakeholders from all parties can get involved. Reduced system development time and cost. (Fu, Bastani & Yen, 2008.)	Not very quick to develop for complex systems. (Fu, Bastani and Yen, 2008.) Requires a developer to build.
3	Rapid prototyping	Prototypes are quick to make and inexpensive. Different design options can be evaluated and iterated quickly. Improves the likelihood of finding a solution to meet user's needs. Unpromising design directions can be cut off to save time and money. (Sears and Jacko, 2009.) Different alternatives available with varying amount of complexity, which can be used in same project. Helps in creating more usable products. Speeds up the development. Increases Customer satisfaction. (Jones & Richey, 2000.) Almost no restrictions on who is able to make the prototypes.	False implication that the product development is also going to be rapid (Sears and Jacko, 2009). Too high expectations for how ready the system is. Lack of mechanisms for requirements traceability. (Fu, Bastani and Yen, 2008.)
4	Card sorting	Simple to understand and do, and therefore usable with most stakeholders. Low-tech, the cards can be taken and used almost anywhere. Amenable to automated tool support. (Maiden, 2009.) Performs well in terms of speed and quantity of elicited information (Upchurch, Rugg & Kitchenham, 2001).	Not easily amenable to statistical analysis. (Upchurch, Rugg & Kitchenham, 2001.) Stakeholders must be sufficiently familiar with the system items. The system must have suitable and sufficient semantic spread across the domain. (Maiden, 2009.)
4	Laddering	Useful when identifying stakeholders' motives and knowledge (Corbridge et al., 1994). Useful in eliciting explanations of technical or subjective terms (Upchurch, Rugg & Kitchenham, 2001).	Repetitive questions can make the interview become exhausting. "Why" questions may get the interview to a too abstract level. Questions may become too personal. (Veludo-de-Oliveira et al., 2006.)
4	Protocol analysis	One of the best techniques to examine the interaction among usability, use experience and ease of use. User's verbal descriptions of actions can reveal usability problems and features that bring Customer dissatisfaction. Does not need a large sample size due to the richness of the collected data. (Benbunan-Fich, 2001.)	Users may describe their actions inconsistently to their recorded behaviour. Lack of common procedures, such as presence or absence of the researcher in the session, the type of instructions provided and the measures for maintaining experimental control (Cabello & Hora, 2002).
4	Repertory grids	Numeric values can be analysed using various statistical approaches.	Complex systems are difficult to represent in a nominal matrix.

Cat.	Method name	Pros	Cons
		Software for further statistical analysis are readily available. (Upchurch, Rugg & Kitchenham, 2001.) Useful in solving disputes and identifying agreements or disagreements between the stakeholder groups in an early phase of software development (Laplante, 2009).	(Upchurch, Rugg & Kitchenham, 2001.)
5	Conversation analysis	Records of naturally occurring interactions improve understanding of human thought and action (Sommerville et al. 1993).	Very labour intensive. Can only be applied on late phase of software development. (Goguen & Linde, 1993.)
5	Ethnographic observation	Useful when addressing contextual factors such as usability. Useful in finding out interactions between users. Effective in when the need for a new system is in correcting existing problems and improving processes. Effective in identifying social patterns and relationships between users. (Aurum & Wohlin, 2006.)	Takes place in user's location (Sommerville et al. 1993). A lengthy process. Produces a large amount of data. Communicating the results and abstracting details of one situation into a design principle is not straightforward. Requires a skilled ethnographer. (Viller & Sommerville, 1999.)
5	Interaction analysis	Useful in discovering details of non-verbal interaction in real work environments (Goguen & Linde, 1993).	Very labour intensive. Can only be applied on late phase of software development. (Goguen & Linde, 1993.)

3.3 Selected methods

If asked to select one method that appears the most in the referenced literature that compare different requirements elicitation methods, Interviews takes the lead, and for a good reason. The different types of interview options and the variability of it makes it suitable for most contexts and studying different kinds of users.

Protocol Analysis is a very commonly used method for usability testing to study already existing systems and it can help detect problems in real interaction situations that are not necessarily revealed by Interviews. The two methods, therefore, may complement each other in defining user requirements. Furthermore, the Protocol Analysis does not necessarily need a large sample size due to the richness of the collected data. As only a limited number of users were available for this study, the method seemed suitable.

3.4 Interviews

Interviews is a conversational method which is considered easy and effective for idea sharing and expressing needs between analysts and stakeholders. It includes face to face conversations with one or two people asking questions and documenting the results which finally lead to requirements. (Yousuf & Asger, 2015.)

Interviews are of 3 types: Structured, Semi-Structured, and Unstructured. The goal of first two is to acquire quantitative data, whereas the third method points towards understanding user expectations through open-ended debates with the stakeholders and acquiring qualitative data (Arif, Khan & Gahyyur, 2010).

Structured Interviews are the most formal type with a set of predefined questions that are asked from the stakeholder. Zowghi and Coulin (2005) consider it an effective technique but claim that it does not allow generation of new ideas. On the other hand, Hickey and Davis (2003) argue that the method is primarily used to surface new information or uncover conflicts.

Semi-structured Interviews are a combination of predefined and unplanned questions. Here the interviewer may ask further defining questions and share and discuss ideas. Unstructured Interviews are informal Interviews containing unplanned questions. It is an open discussion between analysts and stakeholders producing qualitative data. (Yousuf & Asger, 2015.)

Yousuf and Asger (2015) note that Interviews are good with complex topics that require dialogue, and in eliciting detailed requirements. On the other hand, only a limited number of people can be reached, and arranging the Interviews can be effortful and time-consuming.

One problem with Interviews according to Goguen and Linde (1992) is that people know how to do many things that they cannot describe. It is a commonplace in ethnography that people's descriptions of how they weave a basket or choose a chief or write a program, bear a complex and opaque relation to how they can be seen to do these things when they are observed. This problem is so familiar that is has a nickname in social science: *the say-do problem*; also, philosophers speak of *tacit knowledge*.

End-users are usually happy to describe their work and the difficulties they face. Interviews might be a good tool to get to know and meet Customers and therefore later contacting them with a lower threshold.

Interviews were selected for this case, because the processes in which the different users utilise the system vary, and there are multiple types of internal users and Customer users. Semi-structured Interviews give a good overview of all the different users and what they are utilising the system for. Interviewing Customer users could engage users and give them trust for future activities.

3.5 Protocol analysis

The Protocol Analysis method consists of asking the users to perform tasks and think aloud as they work. The process of verbalisation is claimed to reveal assumptions, inferences, misconceptions and difficulties that the users face while solving problems and performing tasks. (Benbunan-Fich, 2001.) Furthermore, the user thinking aloud provides the observer with insights into the cognitive processes used to perform the task (Nuseibeh & Easterbrook, 2000).

According to Benbunan-Fich (2001), Protocol Analysis is an appropriate method for the research of process tracing, knowledge acquisition, model formulation, and decision-making behaviour, and also has been used widely in information systems literature for systems development tasks, model formulation for decision support systems, and for usability studies of computer-based systems and interfaces.

Protocol Analysis is based on the direct observation of a real interaction between the user and the system. The users are instructed to give a running commentary on what they are doing, what problems they are facing, and other task-related thoughts. The user's speech, keystrokes and actions in the system are recorded. The recordings are then transcribed into text and divided into segments, which are then further assigned into different categories. (Cabello & Hora, 2002.)

Typically, only a small group of users is required for Protocol Analysis to yield important results. A sample of five users can uncover 80% of the site-level usability problems. (Nielsen, 2000.)

Protocol Analysis of users interacting with a system can answer the following research questions: how easy the system is to learn and to use, how flexible is it to accommodate different types of interaction styles or users, and why users form specific opinions or attitudes towards a system (Benbunan-Fich, 2001).

Pros and cons that were found of the Protocol Analysis were as follows:

- Protocol Analysis seems to be one of the best techniques to examine the user's interaction with the system from the perspective of objective usability, direct experience and perceived ease of use.
- Users' verbal descriptions of their actions can reveal specific usability problems and features that elicit negative opinions or Customer dissatisfaction.
- Protocol Analysis offers a wealth of information that is generally not available through other methods, and due to the richness of the data collected, protocol analyses do not need to be conducted over a large sample of users. (Benbunan-Fich, 2001.)
- Protocol Analysis provides a tool that allows the researcher to identify information that expert performers do not know how to verbalise while they show their behaviour in the system (Ericsson, 2006).
- When a user is asked how they did the task after an experiment, they sometimes describe their actions and experiences in a way that that is inconsistent with their recorded behaviour (Ericsson & Simon, 1993).

- There is a lack of common procedure for Protocol Analysis, with different researchers using procedures that differ, such as the presence or absence of the researcher during the session, the use of thinking-aloud practice exercises, the type of instructions provided, and the measures for maintaining experimental control (Cabello & Hora, 2002).
- One problem with Protocol Analysis is that it requires significant time and effort due to resampling and transcribing the many hours of tapes recorded of the sessions (Cabello & Hora, 2002).

In the case, where the researcher and participant are not physically located in the same space, requires the researcher to be actively present in the session as the facilitator. If we leave out the recording of keystrokes, Protocol Analysis can be conducted by using basic video conferencing tools, for an example Microsoft Teams, and by utilising its recording possibility, there will be no resampling step.

Protocol Analysis was selected as another method to be implemented in this case as only a small group of users would be needed as participants, the method could be taken through in a rather short period of time. Secondly, as Benbunan-Fich (2001) notes, it seems to be one of the best techniques to examine the user's interaction with the system from to evaluate its usability, user experience and perceived ease of use.

4 METHOD

In this chapter we firstly summarise the research questions. We explain what is meant by a Returns management process, what the current Return system is, its purpose, challenges, and plans for the system upgrade. Then we describe the procedure how the selected user requirement elicitation methods, Interviews and Protocol Analysis were executed and how the gathered data was then processed and analysed.

4.1 Research questions

The goal in this study is to compare the outcomes of two user requirement elicitation methods – how did they differ, which one we would say were better for finding out user requirements if the return system is upgraded and which of the methods yielded more high priority user requirements.

4.2 Returns management system upgrade

Returns management is the supply chain management process by which activities associated with return, reverse logistics, gatekeeping, and avoidance are managed within the firm and across key members of the supply chain (Rogers, Lambert, Keely and García-Dastugue, 2002).

For companies in the hardware manufacturing business, the Customers' need to return a faulty product is a situation, which will come up eventually, and which the Customer can view as unavoidable and a burden. A system that is easy to use and learn can ease the feeling of the "necessary evil". When the whole return process is working efficiently, the Customer can rather see that they have been treated fairly and correctly and can possibly improve and adjust their own processes accordingly.

The current Return system was development in 2009, and since then the processes surrounding the system have been improved a great deal. A few new features have been made to the system, and a lot of usability issues have been fixed along the years, but the basis is still the same, and the implementation of a new system is inevitable at some point. The Return system was developed at a time when there was no need to handle big masses of devices, but since then the material flows have grown notably. In the return process for tens of devices, there can be manual phases, but when reporting by the hundreds or thousands, the manual phases must be minimised, or the outcomes of human error rise to be significant.

A system that would better answer to the users' needs would advances for both the Customer and the Company. Higher usability would encourage the Customer to report the fault in more detail, the Company would gain more information about the causes for returns and information for product development. Finally, when the returned devices would be handled quicker, time and money would be saved for the Company as well as the Customer who would get their repaired or replacing devices delivered sooner.

There are also environmental and sustainability issues to be considered in the returns management process. The Return system can be seen as a service product. If it is not functioning, Customers may see that it is not providing them any value. Dissatisfaction to the product will lead to discontinuation to the use of the system. This can have two outcomes that can be considered from an environmental point of view. First scenario is that none of the devices will be returned to be repaired and therefore none of the devices or parts will be re-used. Other is that the Customer returns all the non-functioning devices in one large batch, the Company goes through them and only then scraps the irreparable devices, when the shipping of those devices could have been avoided.

This thesis will act as a preliminary study, aiming to find out the user requirements and the main improvement needs and areas according to different users of the Return system. The study can be used as a starting point when upgrading the system.

4.3 Returns management process

The Company manufactures devices for the use of other organisations. The Customer company takes care of the installation of the devices to end-customer locations. In a situation where a device is broken, the Customer uses the Company's returns management system to handle the return of the faulty device.

The personnel currently involved in the returns management process work in the following roles: Customer's field technician, warehouse or office personnel, project manager, Company's logistics coordinator, repair, hardware testing and customer project management.

A common scenario is that the person performing the installation or a maintenance operation on the field, notices that a device is broken. From there on forward, the process goes as follows:

1. The Customer's field technician uninstalls the device and attaches a fault ticket sticker that contains a unique ticket number to the broken device.
2. The technician describes the fault (i.e., the reason of return) to the ticket as a code or in free writing if there is no matching code.
3. The ticketed devices are transported to the Customer warehouse, where Customer's personnel report the devices to the Return system based on the information on the tickets, usually in a larger batch.
4. In the system, the devices are selected to be sent and a packing list is printed out and included in the package.
5. Customer opens a collection request form from the system.
6. Customer fills the form and sends it by email for the Company.
7. Company orders the transportation and replies to the email by sending a waybill and a tracking ID for the delivery.
8. Customer prints out and attaches the tracking IDs to each package in the delivery.
9. When the devices arrive to the Company's warehouse, Company personnel perform the incoming material inspection.
10. In the inspection, the devices are counted to an Excel file.
11. Company Logistics Coordinators create repair order rows for the devices to the Company ERP system. There are two rows for each device in the delivery: a replace and a repair row. These are filled based on what is done for each device.
12. The Repair personnel repair or scrap the devices and fill in the corresponding rows in the ERP system to report of material usage.
13. Repair reports the actions done and the status of each device to the Return system.
14. Repair downloads a CSV export of all the devices in the delivery from the Return system and organises it by the scrapped and repaired devices. The list is used to compare and check the number of devices in the ERP system, what the Customer has reported and the devices counted in incoming material inspection.
15. From the CSV file, Repair copy-pastes all the serial numbers of scrapped and repaired devices to "Device state mass update" fields in the system to update the devices' states to "Scrapped" and "Returner to Customer".
16. Repair sends the processed CSV export by email for the Logistics Coordinator and the Customer as a confirmation of the work done.
17. Logistics Coordinator creates new order rows to the ERP system for the devices that will replace the scrapped devices.
18. Logistics Coordinator makes a collection request for the device packages and informs Repair when the devices can be sent back for the Customer.

During the process, both the Customer and the Company can monitor the devices' states in the system and print device information out as CSV reports. The CSV report is used for follow-up and reporting of devices in the repair cycle and in some cases exporting the device information to Customer's other systems.

4.4 Procedure

A total of eight sessions were conducted as Microsoft Teams meetings that were facilitated by the researcher and recorded. Each session took approximately 1 hour and it included both of the requirements elicitation methods, Interviews and Protocol Analysis. Four of the sessions had the Interviews method first and the Protocol Analysis second, and four had the Protocol Analysis first and the Interviews second. A small gift was awarded for each user for participating in the study.

The Interviews were semi-formal and lasted about 30 minutes. The Interview outline and questions are shown in *Appendix 1 Interview structure and questions*. The questions asked were often accompanied by follow-up questions. The questions were slightly altered based on whether the participant was a Customer or a Company user and depending on their role.

In the Protocol Analysis sessions, the tasks to be performed were sent for the participants by email, a few minutes prior to the meeting. In the beginning of each session, the participants were instructed that it was a part of this method that they speak out loud what they are thinking and doing, even if it sometimes feels obvious or odd. They were further asked what they were thinking in times they ran into problems and were quiet. The Protocol Analysis task sheet included the following instructions and tasks structure (only main tasks included):

1. A picture and required information of a broken device to be reported into the return system.
2. Logging in to the system.
3. Reporting the given information to the Return form.
4. Marking the broken device to be scrapped.
5. Marking the device to be ready for shipping for the Company next Friday.
6. Printing out a Packing list.
7. Opening a Pick-up request form, filling and saving it for sending for the Company.
8. Seeing what Company's Repair has commented about some previously reported device.
9. Printing a report of all devices that has been returned with a certain return reason code.
10. Lastly, the participants were asked for any comments that come to mind about the system.

The questions for the Interviews and the structure for the Interviews sessions can be seen from *Appendix 1*. After all the sessions were held, we started the analysis of the recorded material. Issues with the process or the system were picked up from the participants' vocal input or from their actions while using the system. Then we listed the problems that the issues might lead to or were related to, and then derived the user requirement based on how a problem could possibly be solved. The issues, problems and requirements found with both of the methods were arranged into the following categories:

1. *Process* category focuses mainly on finding out the processes surrounding the system. What happens and what has to be done before a device can be reported to the system?
2. *General use and usability* category collects issues related to for example ease of use and layout of the items in the user interface.
3. *Reporting and follow-up* category is related to tasks that are done in the system after the devices are returned and processed.
4. *Integrations* category deals with tasks that require the use or information from other systems to successfully complete the process.
5. *Information security* category collects issues related to safety and privacy of critical data or bugs in the system.

These were roughly the categories when hosting the Interviews and Protocol Analysis sessions but were finally set and decided on in course of organising and grouping the requirements into tables seen in *Appendix 2* and *Appendix 3*. For an example, in Interviews, when asking questions of for example about *General use and usability,* a follow up question could bring up a problem related to another category. Or in Protocol Analysis when performing a *Reporting and follow up* task, a usability issue could arise. In the Protocol Analysis there were no initial tasks in the *Integrations* or *Information security* categories. Problems related to these categories were gathered by deriving them from the users' input and actions while they were performing the tasks. Consequently, some requirements could be related to multiple categories. The most suitable category was selected on the basis whether there already were problems and requirements in a category that were related to the requirement under evaluation. This way the final category could be decided for a group of requirements.

4.5 Data processing and analysis

The found problems and the user requirements derived from the Interviews and Protocol Analysis methods were arranged into tables that can be found from the *Appendix 2* and *Appendix 3*. The tables have the following columns:

- The *User* column indicates the number of each participant as presented in Chapter 5.1 *Participant background information.*

- The *Interview answer/Thinking aloud* column has the quote from Interview or Protocol Analysis in which the participant has expressed an issue with the system or the process. In case of the Protocol Analysis this column may also have an issue we noted related to the user's action.
- The *Problem* column describes to what problems or error the issue may lead to.
- The *Requirement* column has the suggested solution to the problem in a form of a user requirement.

When forming the problems from the Interviews answers or from the user input from Protocol Analysis, we tried to identify all the challenges related to a specific statement. How other users in different roles would see the issue or how does it affect to, for example on the information flows on the Company's point of view. Deriving the user requirements from the problems was based on the researcher's expertise and knowledge of the system, its users, and the returns process as a whole.

The following table shows examples of how problems were derived from the users' input and how the problems were then further analysed and user requirements were suggested based on them. The examples contain three different cases of first identifying an issue, then listing the problems that might lead to and then the user requirement(s).

TABLE 2 Examples of derived problems and requirements

User	Interview answer/Thinking aloud	Problem	Requirement
3, 4, 8	Fault is written to the sticker on the field with a pen.	Using a pen leads to different kinds of user errors. Stickers have to be carried around.	The fault must be possible to be reported to the system in the field with a phone or a tablet.
1, 3	The installer's observations in the field could be reported in more detail. For example, what is shown on the display. How do I know if the red light is on or if the button works. Does the Company think that we power on the device here at the warehouse and investigate? Filling in the fault ticket sticker is one step more. It would be good if it was done in electronic format, so that the fault description would not have to be scribbled in the field.	The fault cannot be described with nearly as much detail when the device is not powered. Whether a device can be powered on is useful information by itself. Adding a fault sticker to the meter is one extra step to the process and separate action outside the system.	The fault must be described and entered to the system in the field by default.
5	We do not need to print Collection requests because we have a Warehousing service from the Company. The devices are already there.	The system expects that all users go through the process the same way. Not marking the devices for delivery leads to devices not changing state in the system until it is done in repair.	The system must indicate what information the user is expected to give in different situations. The system must change the devices' state automatically, after sufficient information have been given of it.

On the first row, some participants said that “Fault is written to the sticker on the field with a pen”. This was considered an issue due to the different problems the practice may and has led to. For example, during the return process, several people have to later read the person’s handwriting, and while carrying a pen is not usually an issue, carrying around a batch A4 size sticker sheets has been. Stickers can be easily forgotten at the office and can be tricky to remove from a device. Therefore, we concluded that both of these problems would be solved if the fault was reported to the system already on the field with a phone or a tablet, that the persons reporting the devices are more likely to already have with them.

The second row is an example of where different kinds of issues led to the same requirement. The reported issues here were different, but as the suggestion for the user requirement is the same, they were grouped. On the last row, there is an example of what a participant said while performing a Protocol Analysis task. Due to the issue that the Customer does not print out a Collection request, the devices’ state does not change to “Sent to Repair”. One user requirement we think would solve the problem is that the system would know what the sufficient information is that this Customer must give, and then the state would change automatically.

5 RESULTS

In this chapter, we firstly describe the participants' background information. Then we present the results of the two requirements elicitation methods Interviews and Protocol Analysis. The counted frequencies of problems, problems per participant and elicited user requirements are shown in tables per category. Lastly, we describe what types of problems and requirements were gathered with the methods to the different categories.

5.1 Participants

The following table lists the background information of the 8 participants in this study. All of the users know and have used the system before. All the participants marked Customer users are from different companies. The numbering in the first column is used later in this study to identify the participants.

TABLE 3 Participant background information

User	Role	Frequency of use	Tasks
1	Company user: Customer project manager	Monthly	Exporting reports of Customer's devices. Processing the files and creating figures of the data.
2	Company user: Production supervisor	Monthly	Receival and sending of devices in the repair process.
3	Customer user: Subcontractor at the warehouse	Weekly	Reporting faulty devices to the system that come from the field.
4	Company user: Hardware testing	Monthly	Testing faulty devices that are sent for investigation. Updating information for reported devices.
5	Customer user: Management	Monthly	Reporting devices infrequently. Receives reports from Company project manager.
6	Company user: Repair	Daily	Receiving, testing and configuring devices that are in the repair process. Creating reports or repaired and scrapped devices.
7	Company user: Repair	Daily	Receiving, testing and configuring devices that are in the repair process. Creating reports or repaired and scrapped devices.
8	Customer user: Management	Monthly	Reporting devices infrequently. Receives reports from Company project manager.

5.2 Frequency of problems

The following table lists the problems that were found from the system by using the two methods. The "All" columns indicate the number of found problems per category and the "Unique" column has the same problems grouped into one.

TABLE 4 Number of problems found from the system

Category	Interview all	Interview unique	Protocol Analysis all	Protocol analysis unique
Process	37	34	15	14
General use and usability	35	35	47	47
Reporting and follow-up	10	10	4	4
Integrations	27	24	9	8
Information security	13	12	15	11
Total	122	115	90	84

5.3 Frequency of problems per participant

The following table lists the problems in the system found with the Interviews method per participant.

TABLE 5 Number of problems found per user from Interviews

User	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Process	18	5	18	9	11	10	0	6
General use and usability	7	3	2	14	8	5	11	15
Reporting and follow-up	6	3	0	3	0	4	4	1
Integrations	9	7	2	3	1	8	5	4
Information security	2	2	1	4	0	2	2	6
Total	42	20	23	33	20	29	22	32

The following table lists the problems in the system found with the Protocol Analysis method per participant. The total counts in this table are large because of grouping similar answers that have the same identified problems. This way all the problems derived from a single answer were counted for all users with a similar answer.

TABLE 6 Number of problems found per user from Protocol Analysis

User	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Process	4	4	2	2	4	7	4	1
General use and usability	26	14	10	23	0	3	1	14
Reporting and follow-up	0	0	0	2	2	2	0	0
Integrations	0	0	2	0	0	7	5	0
Information security	10	6	8	7	1	3	1	3
Total	40	24	22	34	7	22	11	18

5.4 Frequency of requirements

The following table lists the requirements that were derived from the found problems by using the two methods. The "All" columns show the number of elicited requirements per category and the "Unique" column has the same requirements grouped into one. The total number of unique requirements in the "Interview unique" and "Protocol Analysis unique" columns here is lower than the actual

count of the numbers in the column, because the numbers that are on the category rows is the count of unique requirements only in that category. The *Total* numbers here represent the unique requirements in the whole mass and therefore is smaller.

TABLE 7 Number of elicited requirements per category

Category	Interview all	Interview unique	Protocol analysis all	Protocol analysis unique
Process	30	22	15	10
General use and usability	36	30	50	37
Reporting and follow-up	11	5	6	6
Integrations	38	24	7	6
Information security	18	16	14	11
Total	133	82	92	63

5.5 Types of problems and requirements

When comparing the amounts, Interviews had more requirements in all categories except in *General use and usability*, and *Information security*. Of the requirements from Protocol Analysis, more than half were in the *General use and usability* category. As the method is generally used to assess the usability of a system, this was expected. The *General use and usability* category requirements gathered from the Interviews, although fewer in numbers, were related to things that the participants remembered well to be difficult to do in the system.

To the *Process* category we gathered more requirements from the Interviews, as we asked direct questions about it. From the Protocol Analysis, problems that were related to things that the participants said that they usually do, or do not do in the system, or what they have been instructed to report to the system were listed to the *Process* category.

The *Reporting and follow up* category had the smallest number of elicited user requirements as from Interviews we got only 10 and from Protocol Analysis only 5 requirements. In this category the requirements were related to participants missing ready processed reports of the faults and returned devices, and on the other hand, needing more space where to fill in more detailed reports of possible fault causes. From the Protocol Analysis it was seen that Company users spend a lot of time processing the CSV export, available for download in the system, for different needs.

To the *Integrations* category we recorded all problems where the participants needed the help or would have benefited of the use of some other system to do their tasks related to the returns process. Some of these could have been

added to the *Process* category as well, but as it was seen that adding some levels of integration between different system would help in solving the problem, they we categorised under *Integrations*.

The number of requirements gathered to the *Information security* category were almost the same from Interviews and Protocol Analysis with a couple more requirements to the Protocol Analysis. Protocol Analysis proved to be an efficient tool for finding out bugs (~10) from the system. Some users specifically pointed them out while using the system, and some could be noted while the participants were performing the tasks. The bug related requirements were mainly categorised under *Information security*. Problems and requirements gathered from Interviews that we recorded to *Information security* category were related to, for example, the complexity requirements for user credentials and automatic logging out of the system.

The tables in *Appendix 2* and *Appendix 3* containing the issues, problems and user requirements elicited to the *Integrations* and *Information security* categories were omitted from this study due to company confidential information. All the information in the tables was however analysed and processed similarly to other categories.

6 DISCUSSION

The Interviews and Protocol Analysis user requirement elicitation methods proved to be good choices for finding out problems and development issues from the system and for gathering a list of user requirements to be used in upgrading the return system. During the implementation of the methods, a number of points emerged where the system could be improved. For an example, as the system is not designed for the current larger number of devices, there were shortcomings in the flexibility of the system's activity flows causing manual, repetitive tasks, information copying, and cognitive load on the users along the whole process.

There were also good features brought up by the users, for example, the main device return window is simple and consistent due to that it is similar for each user and device type. This study however focuses on the aspects of the return system that require development, in other words, we approach the subject from the users' problems point of view. The study does not take a position on whether the system already meets the presented requirements on some parts, but merely suggests the user requirements that we see would be a solution to the problems seen in users' actions and statements.

The first research question presented in this thesis was "*How did the outcomes of the implemented user requirement elicitation methods differ?*". Of the implemented methods, Interviews yielded more user requirements. With the Protocol Analysis method, we gathered a wide range of requirements related to the usability of the system. Also, illogical behaviour within the system and bugs were reported that would have not come up in the Interviews. The *General use and usability* category requirements gathered from the Interviews, although fewer in number than from Protocol Analysis, were related to things that the participants remembered well to be difficult to do in the system. If the gathered requirements are later prioritised, many of these may stand out as higher priority requirements.

In both Interviews and Protocol Analysis, there were approximately the same number (~10) of system feature update proposals from participants that rose from an actual user need.

In Chapter 2.1 we listed consequences for when the user requirements for a system are incorrect according to Kotonya and Sommerville (1998). The first

consequence mentioned is increased costs. It was seen in Protocol Analysis that all participants struggled with the large amount of manual work in different steps of the return process. This leads to the whole reporting process taking a lot of both Customers' and Company's time. From Interviews we got many requirements to the *Integrations* category (24, compared to 6 from Protocol Analysis) on how parts of the process could be simplified and shortened by adding integrations to other systems the participants are using. One participant Customer has already simplified their processes surrounding the return system. They report the faulty devices on the field, as suggested in the elicited user requirements, and do not mark devices for delivery in the system or print out pick-up requests. This saves time for the Customer but as the system does not support this process, adds more manual work for the Company's repair.

The second research problem was "*Which of the implemented user requirement elicitation methods proved to be better considering the returns management system upgrade?*". The answer to the question, based on this research, is that they are equally good methods. We gathered a lot of usable material in the form of raw user requirements as well as insight to both the Customers' processes and the Company users' needs. As the gathered requirements were different, the methods were seen as complementary. It will depend on the next steps that are taken with the system upgrade, whether the focus is on fixing the usability issues and bugs on the current system found with Protocol Analysis or by completely renewing the system starting with the Process category requirements elicited with the Interviews method.

The final research problem was "*Which of the implemented user requirement elicitation methods yielded more higher priority user requirements?*". A *priority* is an attribute of a requirement which should be the result of the activity called requirements prioritisation. As stated in Chapter 2.4, requirements prioritisation requires complex context-specific decision-making and must be performed iteratively in many phases during the development work. (Lehtola, Kauppinen & Kujala, 2004). Thus, we will not be able to comprehensively answer this question. However, based on the results, some conclusions can be drawn from the number of unique requirements, and if there were issues that were noted by several users. The Interviews method yielded more requirements where more than 3 (of the 8) participants expressed an issue that were related to the same requirement. In Interviews there were 29 and with Protocol Analysis there were 19 requirements that were derived from issues noted by more than 3 participants, that can be therefore considered as higher priority in requirements prioritisation.

In Chapter 2.4 we described three points of view that must be considered when prioritising requirements: Business, Customers and Implementation. Product management needs high-level information about customer preferences, markets, the company's strategy and resources when they decide which requirements will constitute the basis of the product or release. This information is also needed to decide which of the raw requirements or user needs gathered should be the priority and evolved further. (Lehtola, Kauppinen & Kujala, 2004.)

7 CONCLUSION

In this study, we have implemented two user requirement elicitation methods, Interviews and Protocol Analysis, processed and analysed the gathered material, and formed it into user requirements. We have compared the elicited user requirements per category and per method. The total of 145 elicited user requirements are available for the development uses of the Return system in question.

The largest number of user requirements was elicited with the Interviews method from the participants who the researcher was previously acquainted with and who seemed most at ease to express their opinions. As Zowghi and Coulin (2005) state, Interviews are essentially human based social activities, and they are inherently informal and their effectiveness depends greatly on the quality of interaction between the participants. Altogether, the Interviews method provided an efficient way to collect large amounts of data quickly.

It proved to be a good decision to have different kinds of users (as described in Chapter 5.1) involved in the study for finding out the following issues:

- What parts of the system different users utilise.
- How the development wishes and what they see important differ between users.
- What are the surrounding processes that in involve the system and how they differ between organisations.
- How the frequency of use affects the results between the methods.

To cover the functions of the whole process, it would have been beneficial to have users in all roles as participants, for an example, a Company employee who works with the ERP system, and not just the users directly utilising the Return system.

In the case of expert users of the system, Protocol Analysis can be a good tool for finding out how they utilise the system efficiently. Ericsson (2006) states that Protocol Analysis provides a tool that allows the researcher to identify information that expert performers do not know how to verbalise while they show their behaviour in the system. But as the idea of Protocol Analysis is to think

aloud, without that train of thought, the researcher is left with only the video of the expert clicking quickly in the system.

According to Zowghi and Coulin (2005), minor steps performed frequently and repetitively are often taken for granted by the users and may not be explained and subsequently recorded as part of the process. This was seen in the Protocol Analysis sessions conducted in this study, as there were difficulties to get the users talking who use the system in their work in a daily basis and do the given or similar tasks in the system routinely. They went through the tasks quickly as usual, and a large part of their thinking aloud were more or less "I do it like this" and "Then I click here". Much more data were gathered from users who use the system rarely, had some troubles using it, and were not accustomed to the flaws and specialities of the system. On the other hand, one of the daily users provided a lot of information in the Interviews method through their comprehensive knowledge of the process and other mentioned more usability issues in the Interviews than while using the system in Protocol Analysis.

The above example shows that remembering issues with the system in the Interviews might yield to less but higher priority requirements than running into problems during Protocol Analysis. In Protocol Analysis, it could be the first time a participant is using the system and have yet to find the right way to do a task. It of course beneficial to catch these usability issues affecting system's learnability and efficiency. Although, when a participant mentions the same issue in Interviews, they may have tried to solve the problem multiple times but have not found a solution or a workaround.

Cabello and Hora (2002) criticized Protocol Analysis for the lack of common procedure, with different researchers using procedures that differ, such as the presence or absence of the researcher during the session, the use of thinking-aloud practice exercises, the type of instructions provided, and the measures for maintaining experimental control. None of these proved an issue and could be solved by what seemed to be fitting for this study. As the sessions were arranged as online meetings, they naturally required the researcher present as the facilitator. Practice exercises were not needed, mainly because the researcher was present, and thus able to ask participants what they were thinking and encourage them in thinking aloud. As the instructions were in the task sheet sent for each participant, and the tasks remained unchanged for each session, we concluded that the experimental control was maintained sufficiently well.

Collecting the issues and problems from the transcribed material from both of the methods' sessions was time-consuming, and required a lot of organising and reorganising, as the author is a novice in requirements elicitation as well as in requirements engineering. We also could not find help or instructions from previous studies on how to form user requirements from the issues that came up in the sessions, so we had to learn by doing along the process of forming the tables in *Appendix 2* and *Appendix 3*. The categories for the requirements were decided on in course of the organising and grouping the requirements into the tables. The categories were based on what seemed to be common for groups of requirements and what felt like the main development areas for the current

system. The categorisation could have been done differently, for an example according to Maguire and Bevan (2002), as they suggest the categories User requirements, Usability requirements and Organisational requirements, presented in Chapter 2.4. This different way of counting the problems and requirements into categories would have made the results tables in Chapter 5 look different. However, the number of problems and requirements per method and per participant would be the same, and comparable even if the categories were something else.

In this study, we have produced a listing of user requirements derived from the problems found from the system based on the method implementation. The next steps would be in evaluating, prioritising and validating the elicited user requirements to be able to produce a good Requirements Specification document. Each requirement should be checked for completeness, relevance, testability, coherency and traceability. We would need to look at the categories: do they support the development of the product in question or would some other type of categorisation be better for that.

The main contribution of this thesis is that it describes the implementation of two user requirement elicitation methods all the way to forming the initial user requirements for system development. By comparing the methods and describing the outcomes, researchers and practitioners may be in a better situation in selecting the best elicitation method to suit their needs, whether it would be Interviews, Protocol Analysis, or something else described in the *Elicitation method pros and cons* table in Chapter 3. The study shows that combining different elicitation methods, researchers get more sufficient and extensive information on the processes related to the system, and the different needs and problems the stakeholders are facing.

REFERENCES

Arif, S., Khan, Q., & Gahyyur, S. A. K. (2010). Requirement Engineering Processes, Tools/Technologies, & Methodologies. *International Journal of Reviews in Computing*, ISSN: 2076–3328, Vol.2.

Carrizo, D., Dieste, O., & Juristo, N. (2014). Systematizing requirements elicitation technique selection. *Information and Software Technology, 56*. 644–669.

Corbridge, B., Rugg, G., Major, N. P., Shadbolt, N. R. , & Burton, A. M. (1994). Laddering - technique and tool use in knowledge acquisition. *Knowledge Acquisition, 6*(3), 315–341.

Coughlan, J., & Macredie, R. D. (2002). Effective communication in requirements elicitation: a comparison of methodologies. *Requirements Engineering, 7*(2), 47–60.

Davis, A. (1992). Operational Prototyping: A New Development Approach. *Software, 9*(5), 70–78.

Diehl, M. & Stroebe, W. (1987). Productivity loss in brainstorming groups: Toward the solution of a riddle. *Journal of Personality and Social Psychology, 53*(3), 497–509.

Duggan, E. W. & Thachenkary C. S. (2003). Higher Quality Requirements: Supporting Joint Application Development with the Nominal Group Technique. *Information Technology and Management, 4*(4), pp. 391–408.

Ericsson, K. A. & Simon, H. A. (1993). *Protocol Analysis: Verbal Reports As Data*. London: The MIT Press.

Ericsson, K. A. (2006). Protocol Analysis and expert thought: Concurrent verbalizations of thinking during experts' performance on representative tasks. *The Cambridge handbook of expertise and expert performance*, 223–241.

Farinha, C. & Mira da Silva, M. (2009). Focus Groups For Eliciting Requirements In Information Systems Development. *UK Academy for Information Systems Conference Proceedings 2009*, Oxford, UK.

Goguen, J. A. & Jirotka, M. (1994). *Requirements Engineering: Social and Technical Issues*. London: Academic Press.

Goguen, J. A. & Linde C. (1993). Techniques for requirements elicitation. *Proceedings of the IEEE international symposium on requirements engineering,* 152–164.

Heninger, K. L. (1980). Specifying Software Requirements for Complex Systems. New Techniques and Their Applications. *IEEE Trans. on Software Engineering, 6*(1), 2–13.

Hickey, A., Davis, A. M. (2003). Elicitation technique selection: how do experts do it? In: *Proceedings of the 11th IEEE international requirements engineering conference (RE'03)*, Monterey, California.

IEEE Standard 610.12-1990. (1990). *IEEE Standard Glossary of Software Engineering Terminology*.

Jones, C. (1996). *Patterns of software systems failure and success*. London: International Thompson Computer Press.

Jones, T. & Richey, R. (2000). Rapid prototyping methodology in action: A developmental study. *Educational Technology Research and Development*, *2*(48), 63–80.

Kotonya, G. & Sommerville, I. (1998). *Requirements Engineering Processes and Techniques.* New York: John Wiley and Sons.

Lamm, H., & Trommsdorff, G. (1973). Group versus individual performance on tasks requiring ideational proficiency (brainstorming). *European Journal of Social Psychology, Vol. 3*, pp. 361–387.

Laplante, P. A. (2009). *Requirements engineering for software and systems*. Boca Raton: CRC Press.

Lehtola, L., Kauppinen, M., & Kujala, S. (2004). Requirements prioritization challenges in practice. In *International Conference on Product Focused Software Process Improvement*, pp. 497–508.

Maguire, M., & Bevan, N. (2002). User requirements analysis. In *IFIP World Computer Congress*, *Vol. 13*, pp. 133–148.

Maiden, N. (2009). Card sorts to acquire requirements. *IEEE Software, 26*(3), 85–86.

Maiden, N., & Rugg, G. (1996). ACRE: selecting methods for requirements acquisition. *Software Engineering Journal*, *11*(3), 183–192.

Merriam-Webster. (2022). Elicit. In *Merriam-Webster.com dictionary.* https://www.merriam-webster.com/dictionary/elicit.

Nielsen, J. (2000). *Designing Web Usability: The Practice of Simplicity*. Indianapolis: New Riders Publishing.

Nijstad, B. A. & De Dreu, C. K. W. (2002). Creativity and group innovation. *Applied Psychology: An International Review, Vol. 51*, pp. 400–406.

Nuseibeh, B. & Eastbrook, S. (2000). Requirements Engineering: A Roadmap. *Proceedings of the Conference on the Future of Software Engineering*, pp. 35–46.

Okwemba, R. K. (2019). *Requirement elicitation framework for re-engineering diagnostic health care information systems in Kenya.* Kolkata: Exceller Books Global Press.

Pohl, K. (1994). The Three Dimensions of Requirements Engineering: A Framework and its Applications. *Information Systems, Vol. 19*, pp. 243-258.

Rogers D. S., Lambert D. M., Croxton K. L. & García-Dastugue S. J. (2002). The Returns management process. *The International Journal of Logistics Managemen,t 13*(2), 1–18.

Sears, A. & Jacko, J. (2009). *Human-Computer Interaction: Development Process*. Boca Raton: CRC Press.

Shaw, M. & Gaines, B. (1996). Requirements Acquisition. *Software Engineering Journal, 11*(3), 149-165.

Simon, J. (1999). How To Conduct A Focus Group. In J. Simon, *The Wilder Nonprofit Field Guide to Conducting Successful Focus Groups*, pp. 9–34, Saint Paul, Minn: Amherst H. Wilder Foundation.

Sommerville, I. (2005). Integrated requirements engineering: A tutorial. *IEEE software, 22*(1), 16–23.

Sommerville, I. (2000). *Software Engineering*. Harlow, England: Pearson Education.

Sommerville, I., Rodden, T., Sawyer, P., Bentley, R. & Twidale, M. (1993). Integrating ethnography into the requirements engineering process. P*roceedings of the IEEE International Symposium on Requirements Engineering*, pp. 165–173. IEEE.

Sommerville, I., Sawyer, P. (1997). *Requirements engineering – A good practise guide.* West Sussex: John Wiley & Sons Ltd.

Upchurch, L., Rugg, G. & Kitchenham, B. (2001). Using Card Sorts to Elicit Web Page Quality Attributes. *IEEE Software, 18*(4), 84.

Yousuf, M. & Asger, M. (2015). Comparison of Various Requirements Elicitation Techniques. *International Journal of Computer Applications, 116*(4), 8–15.

Veludo-de-Oliveira, T. M., Ikeda, A. A., & Campomar, M. C. (2006). Discussing laddering application by the means-end chain theory. *The Qualitative Report, 11*(4), 626–642.

Viller, S. & Sommerville, I. (1999). Social Analysis in the Requirements Engineering Process: from ethnography to method. In *Proceedings IEEE International Symposium on Requirements Engineering,* pp. 6–13. IEEE.

Wood J. & Silver D. (1989). *Joint application design: how to design quality systems in 40% less time*. New York: John Wiley & Sons Inc.

Yue, T., Briand, L. C., & Labiche, Y. (2011). A systematic review of transformation approaches between user requirements and analysis models. *Requirements engineering, 16*(2), 75–99.

Zowghi, D. & Coulin, C. (2005). Requirements Elicitation: A survey of Techniques, Approaches and Tools. *Engineering and Managing Software Requirements,* pp. 19-46. Berlin: Springer.

APPENDIX 1 INTERVIEW STRUCTURE AND QUESTIONS

***Process* category questions**

1. What job positions the people work in your company that deal with the returns process?

    a. Who writes the information to the fault sticker?
    b. Who transports the faulty devices from the field?
    c. Who handles the receival of the devices coming from the field?
    d. Who reports the devices to the system?

2. Starting from the situation that there is a broken devices that is taken from use on the field, what happens to it then?

    a. Do you attach a fault sticker to it?
    b. How do you report what was wrong with it?
    c. Does the personnel on the field have a tablet or a PDA device with them?
    d. How are the faulty devices transferred from the field?
    e. Are the devices moved again for reporting to the return system?
    f. Do you do the reporting on one PC?
    g. Do you report the devices in batches?
    h. How long would you estimate it takes for you to report that batch to the return system?

3. What happens to devices after they are reported into the system?
4. How do you handle devices that will be scrapped? Do you mark them to the system that they are going to be scrapped?
5. How often do you use the return system?
6. What functions of the system do you use?

General use and usability **category questions**

1. What kind of a device do you have for reporting devices?
2. Do you have a barcode scanner?
3. Is there an external display attached to your computer?
4. Do you think the setup is well suited for using the system? (For example, do you think the display is of appropriate size?)
5. Have you been taught how to use the system?
6. Have you taught anyone how to use the system?
7. What comes to mind of the teaching situation? For example, what kind of questions did the person being taught ask?
8. Is there a function that you have checked from the user manual to see how to do it?
9. Have you had to ask for help using the system? What did it concern? How could the system help you with this?
10. Do you report anything other than whole devices to the system?
11. Is it easy to transfer the data from Fault Sticker to the return system?
12. Has the sticker usually been filled in with the necessary information?
13. Do you know if the person filling the sticker has instructions on how to fill it and explanations for the return reason codes?
14. Is it clear how to report information of the device to the Return Form in the system?
15. Do you check from somewhere which return reason code should be used to report the device into the system?
16. Has it been easy to choose a fitting return reason code?
17. Have you marked the devices for scrapping? Do you know how to do it?
18. Do you know when device warranty is valid or void?

Reporting and follow-up **category related questions**

1. Do you track the status of devices in the system? You think it is easy?
2. Do you think there are enough states? Are the states descriptive?
3. Do you read Repair's/Customers' comments on the returned devices?
4. In what kind of cases are they most interesting?
5. Do you print out lists of devices from the system? For what kind of use?
6. Do you send them for anyone? As such or do you process and modify them first?
7. Do you think the system is missing some kind of reports?

***Integrations* category related questions**

1. Do you utilise the data from the Return system in any other systems?
2. Would it be useful?
3. Do you use any other system, for an example, in the field for reporting related information to?
4. Do you think some more information should be reported about a faulty device in the field?
5. Do you think some more information should be reported about a faulty device in Company Repair?

***Information security* category related questions**

1. Is there any information related to the returns process or asked in the Return system that you would not like to save there?
2. How would you assess the information security of the Return system?
3. What information security related issues do you see about the returns process?
4. How important is security for you when handling the devices?

Feedback

1. Have you had any feedback from Customers/colleagues of the Return system?
2. Have you heard any thoughts about the system from anyone?
3. Would you have any other feedback of the system that you would like to share?

APPENDIX 2 REQUIREMENTS DERIVED FROM INTERVIEWS

TABLE 8 User requirements from Interviews: Process

User	Interview answer	Problem	Requirement
3, 4, 8	Fault is written to the sticker on the field with a pen.	Using a pen leads to different kinds of user errors. Stickers have to be carried around.	The fault must be possible to be reported to the system in the field with a phone or a tablet.
1	We send devices to be scrapped and recycled from the office to production in batches. They are not usually reported to the return system.	Some of these are Customer devices that have been sent to the lab for investigation. The information from the investigations or testing may be lost.	The system must be so simple to use that all devices are reported there. The system must support and encourage Company users to add more information about the fault causes.
1, 6	The return process is quite complex and involves a lot of manual steps and therefore takes a long time. The short process has been made complicated.	Users work with emailing, word processing, PDF reader, and spreadsheet software, as well as ERP, production, Customer information and document management systems regularly to keep the returns process running. Company users have to do device state transitions in the system manually. Adding a fault sticker to the meter is one extra step to the process and separate action outside the system.	The system must support performing all possible steps of the process in the system. The system must change the devices' state automatically, after sufficient information have been given of it. The device must be reported to the system without using the fault stickers.

1, 6	We update the "Repair action" field to "To scrap" or "Fixed" for each device manually. Then we copy-paste the device IDs of all replaced and scrapped devices from the CSV file to the corresponding mass update fields. This will change the device states as "Scrapped" or "Returned to Customer". The information of scrapped devices does not keep up to date. The Customers frequently have devices that have been scrapped, but the system shows that they are in repair. The Customers do not always mark the devices for delivery and then the device states are not correct. Repair then changes the state to "In process" in receival.	Company users have to do device state transitions in the system manually. Some states have to be set in two different places. If the user forgets to update the states, the information flows in the system stop and for example the "Scrapped devices" view does not show anything.	The system must change the devices' state automatically, after sufficient information has been given.
1, 3	The installer's observations in the field could be reported in more detail. For example, what is shown on the display. How do I know if the red light is on or if the button works. Does the Company think that we power on the device here at the warehouse and investigate? Filling in the fault ticket sticker is one step more. It would be good if it was done in electronic format, so that the fault description would not have to be scribbled in the field.	The fault cannot be described with nearly as much detail when the device is not powered. Whether a device can be powered on is useful information by itself. Adding a fault sticker to the meter is one extra step to the process and separate action outside the system.	The fault must be described and entered to the system in the field by default.
3, 4, 5, 6	There was a fault sticker on a device that said, "arrived without description". We instruct to attach the sticker to the device in the field but that had not happened. Sometimes there is only the fault sticker on a device with no information and device has not been entered into the system. We do not use the fault stickers. They would go to scrap and installers would need to carry big piles of them around. The installers do not have the time to put on stickers.	Encouraging sticker use may imply that there is no need to report the device to the system. Attaching stickers to the devices creates waste. Field workers have to carry the sticker sheets around. Adding a fault sticker to the meter is one extra step to the process and separate action outside the system.	The device must be reported to the system without using the fault stickers. To discontinue using fault stickers, the system must allow making a quick check whether a device has already been added to the return system.
2	It is sometimes hard to read the text on the stickers because it is handwriting or in Norwegian.	The stickers are filled usually on the field, sometimes with gloves on and in such environment that impairs handwriting. Writing on a localised sticker encourages to write with the same language, even though some languages are preferred. Preferred language skills cannot be expected from all users.	The device must be reported to the system without using the fault stickers. It must be easy to select a correct fault code so that as little free texts is needed as possible. System must support describing the fault on different languages. The system must have translation tool support for free text fields.

3	The installer has written the fault behind the device with a pencil, I transfer the text to the fault sticker and then write the same information to the return system.	Writing the fault two times during the process means extra manual work and frustrates the user. Copying handwriting to another handwritten text duplicates the mistakes. After the device has been removed from the field, there is no possibility to power it later, and therefore all of the fault codes do not apply.	The fault must be possible to be reported to the system in the field.
1, 3, 5	The installers carry a separate paper that have fault sticker use instructions, return reason codes and fault codes. I doubt that all have that with them. The installers have the error and fault code lists printed out and attached to the back of their tablets. I have the error and fault codes on a separate paper. The list is very long and it is hard to find a suitable code. That is why I put "other" a lot of times. If the right code is not found quickly, we do not start looking.	The fault code drop-down menu in the system is so long that they are provided on paper to help in selecting the codes. The return reason codes and error codes that can appear on device display are in the same drop-down menu. Without the paper, it is hard to select the correct fault or return reason code. When a code is difficult to find, nothing or a wrong code is selected as the reason.	When the user has to select between different items in a menu, the items must be all visible without scrolling. Fault codes must be organised in separate menus by their type. The system must give instructions to help the user in decision making.
1, 4, 5	The fault is not always found from the list and description in writing has to be added. The fault code 999 is used a lot.	If a ready fault option is not found, not all users will write a description by hand. User may select a nearly matching or a wrong fault code so they do not have write to write a description by hand.	It must be easy to select a matching fault code and a return reason code. A fault code must be found for all kinds situations where a device is returned. Fault codes must be organised in separate menus by their type.
1, 2	Reporting information is clear when a whole device is returned. I have had questions about what to do when it is a part of the device. I do not know how to report something else than a whole device.	Confusion about what information to give may lead to not returning or reporting the device at all.	The system must indicate what information the user is expected to give in different situations. The system must guide the user through the return process.
1	There should not be too many return or error codes so that there are too many similar options.	User may not have the time to go through many fault codes and select between them. The return reason codes and error codes that can appear on device display are in the same drop-down menu. Therefore, there are duplicates.	Long drop-down menus must be split by category.
2	I don't remember how it works, if the warranty is not valid then will it be scrapped because the device is still the Customer's property.	Each Customer can have a different warranty procedure and time period that devices must be held on to before scrapping. Users cannot be expected to remember these.	The system must guide the user through the process. The system must inform the user what to do with the device if the warranty is void.
3	User proposes that how about if we could dictate verbally what the situation is.	There can be situations in the field where writing the fault cause is not possible.	The system must support dictating the fault situation to an audio file for the Company repair's use.

3, 4, 5, 8	The devices are reported and returned in batches of 10. During a certain project phase, I had to report batches of 50–100 devices.	Adding the same information multiple times is time consuming.	System must support reporting devices in batches.
8	I have to walk back and forth between the warehouse and my computer when measuring packages and filling in the Pick-up request form.	Ordering a pick-up is time consuming. A laptop computer with barcode reader attached to it is hard to carry around.	User must be able to use the system with a phone or a tablet.
8	I print out the Pick-up request form from the system, fill it by hand when measuring packages and then enter the same information to the form on a computer and send it by email.	Filling the "Pick-up request form" twice is time consuming and prone to error.	User must be able to order a pick-up from the system.

TABLE 9 User requirements from Interviews: General use and usability

User	Interview answer	Problem	Requirement
1	The return form asks for information stating that it is mandatory but not really mandatory to report.	Fields being marked mandatory likely increases the times information is added to those fields. But if the field is really required to be filled, it should not be possible to leave it empty.	The system must inform the user when they leave a mandatory field empty. The system must not allow saving a form with critical fields empty.
2	Return reason or fault code is the most critical thing to have in the fault sticker.	If there is no error code written to the fault sticker, repair must start looking for the return reason from scratch.	The return reason field must be mandatory to fill in the system.
1, 3, 5, 8	It is awkward because I have to alternate between a barcode reader and the tab key on the keyboard that goes to the next field. It is a really tricky extra movement. I do not have a barcode reader, because I use the system rarely. Then I have to enter the serial numbers by hand. If I had a tablet and its camera to use the system with, I would not need a separate barcode reader.	Using a barcode reader and keyboard limits where adding a device to the system can be done because they make the setup harder to move around. Working with keyboard and barcode reader is slow and cumbersome and has working ergonomics issues.	User must be able to use the system with a phone or a tablet. The system must support using a phone's or tablet's camera to read a barcode to the system.
2, 6, 7	If I need help I usually ask from my colleagues. I have tried to look for instructions for Company users on what to write to the repair fault description field and how to do the CSV files but there is none. Creating the CSV files that we send for the Customer was complicated and hard to remember.	Colleagues do not always have the answer and have to guess. Having to ask around for instructions is time consuming.	The system must have user instruction available for all types of users. Fields must have instructions as pop-up info buttons.
5, 7	If there has been a clear fault with the device, we try to describe it as accurately as possible. It is useful if the Customer has described what might be wrong with the device.	There is only one field with erroneous title in the system for fault description and it is not mandatory.	The system must support and encourage users to give more information about the possible and identified fault causes.
8	The system does not give feedback whether I have given all the required information.	User would be willing to help by giving more information but the system does not indicate where to enter that data. This causes the Company not getting all the information of fault causes. The Company users take longer time finding out the reason of return.	System must inform the user what information would be useful in different fault situations.
8	I do not think I use the system the way it is intended. Though the system does not give feedback whether I have filled in all the necessary information.	The user does not know if they are doing everything correctly. The system has a lot of fields of which the user does not know if they should be filled.	The user must be given feedback whether an action in the system was sufficient or incomplete. The system must indicate what information the user is expected to give in different situations.

4	Sometimes I have asked for help because the system does not give feedback if I have entered something wrong.	As there are no instructions for filling information or data validation in the fields, wrong type or inaccurate information can be entered to the system by mistake.	The fields that user fills to the system must have instructions what type of information is required. The system must inform the user if they have entered wrong type of information or if some mandatory fields are left empty.
5, 8	I often check the user manual for the order of marking the devices for delivery and ordering pick-up.	Selecting the devices for delivery and ordering a pick-up is time consuming and prone to error. Marking the devices for delivery has to be done in a specific order in different system windows and it is hard to remember without he manual at hand.	The system must guide the user through the different steps in the process.
5, 6, 7, 8	We train new employees to use the system. Old users train the new users.	There is an official training for Customer users in start of a project. Latter trainings are not controlled by the Company. There are no official trainings or a training agenda for new Company users. Old habits or incomplete processes may be transferred on for the new users.	New user trainings and/or training material must be available that goes through all system features the users is required and expected to use. The system must have a user manual that describes the functions also for the Company users.
1, 4	In the trainings I have given, we go through the return system user manual as the training material.	Reading the user manual is not the same as using the system.	User trainings must include demonstrations that are done in the system.
4	I have never seen the user manual; I did not know that it exists.	When there is no user manual, users may repeat the same errors and some questions are left unasked.	The user manual must be clearly visible and accessible to users.
8	When I start typing numbers to the error code drop-down menu, the menu selects the return reason code starting with that number.	Only one user mentioned this feature as others told they were struggling to find the correct codes from the menu.	The fields in the system must have tips and instructions on their use. Drop-down menus must be searchable.
8	The warranty expires field could turn red when the date has passed for it to pop out better.	If the expired warranty is not noticed, devices are reported and shipped only to be scrapped by the Company.	Device's expired warranty must be highlighted for the user.
4	The layout of the return form is not the best. I usually fill both the Customer's and repair's side, and I am neither.	There are cases where all users are expected to fill in information to "Customer fills" and "Repair fills" side. This is confusing for the user. This leads to users not filling in required information.	The system must have the knowledge which type of user is currently logged in. The system must show only the relevant fields for the device the user is returning.
7	I sometimes have problems with the font. The text is really small. Then if you zoom in, the Close/Minimize/Full Screen buttons on the top of the window disappear.	This is a working ergonomics issue. It is hard for the user to spend the day reading/writing text that is too small.	The user must be able to change the system font size. The UI must be scalable for different display settings and setups.
4	The "Fault cause" field is hard to fill. How do I know if it was the manufacturer or the Company?	A user figuring out who caused the fault is unnecessary and time consuming. They can be only expected to share the information they currently have of the fault.	The system must indicate what information the user is expected to give in different situations.

7	If a Customer has their own serial number for the device, the process takes longer. There is more checking to do and if I send the device back with a wrong number, it will be sent back again.	Customer serial numbering involves extra manual steps to the process and is time-consuming and prone to error.	The system must have the information of Customer's own serial numbering. The system must inform the Company user of the replacing devices' Customer serial number.
5	There are two serial numbers on the device, I have been asked which one or both the user should enter.	The system has many fields that are not expected to be filled by a Customer user. This is time-consuming and may cause the important fields not to be filled.	The system must have the information of Customer's own serial numbering. The system must indicate what information the user is expected to give in different situations.
4	User proposed that there could be fields that are filled automatically when you log in, such as Reporter.	Filling all the fields manually for each device is time-consuming.	The system must not ask the user to fill in information that is already known.
4	Some fields are never filled for most device types.	Asking unnecessary information may cause user leaving necessary fields not filled. Meticulous users spend time filling in unnecessary information.	The system must indicate what information the user is expected to give in different situations.
1, 4, 8	The system is outdated.	The system's age shows as usability issues. No other of today's systems work similarly. Manual steps in the process cause low memorability and human error. Poor reputation and bad experiences of the system causes it not to be used.	The system must work on modern tools. The system must have high usability.
4, 7	The search could be improved. It should be possible to search for the whole device with a device part's serial number.	The search accepts only device serial number and fault sticker ID as the search criteria. The filtering option is hard to find and unintuitive. When the device part serial number cannot be used in the search, its connection to the device and the repair and fault history of it are lost.	The search must have common filtering and sorting options. Users must be able to search with all information given for the reported devices.

TABLE 10 User requirements from Interviews: Reporting and follow-up

User	Interview answer	Problem	Requirement
1	I do reports for the Customers and analysis of what the fault causes were. Then, I download the CSV file and further process it myself. I calculate different percentages and statistics and make pivot tables for internal and Customer use. It is more useful for the Customer if the data is ready and processed.	Sending out raw data CSV files are not useful for comparing figures and follow-up purposes. Processing the CSV files is cumbersome and time-consuming.	The system must offer ready processed reports of selected devices. The user must be able to filter rows of reported devices in the system based on different criteria.
1, 4, 6, 7	Repair's comments on what has been done for the device is a good source of information when the Customer has not given much. It would be a good idea to do more research on the faulty device in repair not just change the device or part of it to a new one. Repair does not describe what has been wrong with the device. Only if the warranty is valid, if it has been scrapped and replaced or has the module been replaced.	Currently the process or the system do not support adding extra information of what was wrong with a devices. Recurring problems can continue without Company noticing.	The system must support and encourage users to give more information about the possible and identified fault causes.
1, 2, 6	When the devices are sent back from repair to the Customer, repair also sends the CSV export files taken from the system as email to the Customer. We inform the logistics coordinators by email that the work is done and attach the created CSV files.	Creating and sending the report is time-consuming. Using Excel and email involves multiple steps outside the system.	The system must send the device follow-up emails for necessary parties after a repair batch is marked processed.
7	If I make a typo to the "Repair fault description" field when processing the CSV export for sending to the Customer, the device row will get deleted and be missing. To avoid that, I compare the amounts in the Customer return form I have in email to the summary list I made in receival to make sure the number of rows match to the number of devices.	Repair writes the Repair Order number to the "Repair fault description" field to identify a device in a specific delivery. This manual work is prone to error. Processing the CSV file by hand is time consuming and prone to error.	The Repair Order information must be added to the system when it is known that the devices are coming to repair. The system must add Repair Order information for each device in a delivery automatically. The system must inform the user if the is a mismatch in device amounts. The system must offer ready processed reports of selected devices.
8	It would be useful if I could check from the system the number of both repaired and replaced devices.	The information is now delivered as Excel sheets in email. This is time consuming and prone to error.	The system must offer ready processed reports of selected devices.
2, 4	If there is a problem device that gets returned again we dig up old information given to it. I use the system for problem solving and see whether the same fault has occurred earlier with other devices.	There is only small space to gather fault history for one device.	The system must collect fault history and give occurrence reports. Fault codes and descriptions must work as a search criteria in device search.

APPENDIX 3 REQUIREMENTS DERIVED FROM PROTOCOL ANALYSIS

TABLE 11 User requirements from Protocol Analysis: Process

User	Thinking aloud	Problem	Requirement
8	We do not report devices to be scrapped. Only the devices we send for repair.	The Company does not get all information of fault causes.	The system must be so simple to use that all devices are reported there.
6, 7	I always fill in ”other” to the “Repair fault code” field. To the “Repair fault description” field, I write the repair order number and for example ”device does not start up, replaced, configured and tested”. If I warranty is void, I write ”Warranty expired, device scrapped”.	Repair does not add the information of what was the final fault cause. When the warranty is void for a device, the fault is not investigated. The Company does not get all information of fault causes.	The system must support a process where the fault is investigated. The system must encourage users to add more information about the fault causes.
2, 6, 7	In some cases, it is unclear for the user when warranty is valid. Customers generally report that the device is under warranty. If the fault has been caused by ”Environment” it is unclear to me if warranty is void or not. I would have to ask my supervisor. I check the warranty procedure from a Customer specific document in our document management system. It says for example, how many years the warranty is valid, and which device type replaces which.	The users should not have to guess whether a warranty is void or valid. This is time consuming and increases human error.	The system must have the warranty information for all devices. The system must be able to tell whether the warranty is valid or void for a device.
1, 2, 3	Notes that the system could not find information to the “Warranty expires” field. The “Warranty expires” field shows only zeroes, even though the “Delivery date” was found for the device.	Checking and searching for information from different systems is manual and time consuming work. The users should not have to guess whether a warranty is void or valid. This is time-consuming and increases human error.	The system must have the warranty information for all devices. The system must be able to tell whether the warranty is valid or void for a device.
4	User did not know that the fields “Repair action” and “Warranty” must be filled for devices to be scrapped.	The user manual tells to fill these fields but not all users have it. Filling information to the “Repair fills” side of the system is unintuitive when the user is not repair personnel.	The system must guide the user through different steps of the process. The system must indicate what information the user is expected to give in different situations.
6	If the checkbox ”Save data from device” on the sticker would be selected we would enter that information to the ”device data” field, but it never is.	This was the only mention of this checkbox on the sticker and field on the system. If the functionality is not used, it is only a waste of space on the sticker. This information could be filled to some “Additional information” field in the system that can be used for other purposes too.	The system must not have fields that are never filled.

1, 5, 6	There are a lot of fields that the user has learned not to fill. We only fill in the fields “Fault sticker number”, “Device serial number”, “Return reason code”, “Fault description”, and “Reported by”. The “Fault cause” field: If the device has been dropped I may put “Installer” here, otherwise nothing.	The system has so many fields that are not mandatory, a Customer has made their own internal instruction on what to fill. A user figuring out who caused the fault is unnecessary and time consuming. They can be only expected to share the information they currently have of the fault.	The system must indicate what information the user is expected to give in different situations. The system must not have fields that are never filled.
5	We do not need to print Collection requests because we have a Warehousing service from the Company. The devices are already there.	The system expects that all users go through the process the same way. Not marking the devices for delivery leads to devices not changing state in the system until it is done in repair.	The system must indicate what information the user is expected to give in different situations. The system must change the devices’ state automatically, after sufficient information have been given of it.
2	No repair order number is given for parts of devices.	If some devices are not reported to the system the Company does not get all information of fault causes.	The system must support reporting and returning all Customer’s devices.

TABLE 12 User requirements from Protocol Analysis: General use and usability

User	Thinking aloud	Problem	Requirement
8	Enters the error code by typing "46" on the keyboard, which was the code given in the task. The correct code is selected directly without scrolling through the drop-down menu.	User has to know about this feature to use it. It is hard to find the correct item from the long drop-down menus.	The fields in the system must have tips and instructions on their use. Long drop-down menus must have a search option.
4, 6	Customers often set the device status as ”Not repaired” and we change it to ”Repaired” or ”Scrapped”. The system asks for a kind of serial number that this device type does not have.	Customer’s use time filling fields that are not required of them and are unnecessary.	The system must indicate what information the user is expected to give in different situations.
1, 3, 8	What repair has filled in what has been done for the device to “Repair fault description” field, is not entirely visible. Does not notice that the field can be enlarged from the corner. Comments that the “Repair fault description” is not very readable because only a small part is visible at first and then you have to know how to make the field larger.	Users sometimes do not bother to read information if they have to do extra work to access it. When searching for information from the “Repair fault description” field, widening the field is cumbersome.	All text written to the fields by users must be visible when the display size allows it.
3, 8	User wonders if “Batch date” means the date when they are going to send the devices. When selecting a date for the delivery the date picker field title is “Batch date”. User calls this “Due date”.	Misleading field names creates confusion and extra checks for the users.	The user interface terms and language must follow common terminology used on the field. The fields in the system must have tips and instructions on their use.
1, 2, 4, 8	User comments that the device table in “Delivery to repair” window is in the wrong order as the newest is the last. User comments that in another table the device with latest date is found from the first row. User has to use the search to find a device because it is not visible at the bottom of the table.	The fact that the latest reported device was not on the first row made the users question all their previous actions in the system and think that they have done something wrong. Looking for missing items is time-consuming and decreases the users’ trust for the system.	Tables that are organised by the date, the latest date must be on first row.
4	After searching for the latest reported device in the “Delivery to repair” window, there is a message saying “Found 1 pcs” but nothing happens. User must be guided to find the device from the bottom of the table.	The search can be used to select the devices for delivery but the window or the message does not say anything about that. If the latest device was on the top row, the user could have seen that the device is now selected.	Users must be informed of system’s automatic actions. The fields in the system must have tips and instructions on their use. Tables that are organised by the date, the latest date must be on first row.
1, 2, 3, 4	User does not know what to put to the “Reference” field in “Delivery to repair” window.	Reference is a misleading title for the field as it is not used in the same context as usually with deliveries. A free text field with no explanations leads to leaving it blank.	The fields in the system must have tips and instructions on their use. The user interface terms and language must follow common terminology used on the field.

2, 4, 8	The “Print” button is hard to find, and user clicks the “View” link to see if it is there. Wonders whether a reported device must be opened with the “View” link first before a delivery list can be printed.	When searching for the printing option, the only clickable item in the window seems to be the “View” link and it draws users attention.	All links and buttons should have styles defined for them that they can be found easily and the user knows what to look for.
8	Only a small part of device table fits to a window.	The window listing all reported devices opens too small by default.	The tables in the system must be visible as the display size allows it.
4	There is no code for device button being broken.	Missing codes lead to the faults not being reported.	The system must have an option for the users to send feedback and report bugs.
4	The return code “display is blank” can mean a lot of things.	This code is an example of a fault that can no longer be recognised when the device is not powered anymore.	The system must separate fault codes for when the device is powered up and when the power is off.
1, 4, 8	Searching reported devices by the fault code in “All returns” window finds nothing.	The search accepts only device serial number and fault sticker ID as the search criteria.	Users must be able to search with all information given for the reported devices.
6	Only device serial number or fault sticker ID can be used to search for devices in “All returns” window.	There are no instructions on how to use the search, e.g., what are the accepted search criteria.	Users must be able to search with all information given for the reported devices. The fields in the system must have tips and instructions on their use.
1, 2	Filtering the search with a return code does nothing when there is no Customer selected.	There are no instructions on how to use the filtering. The filtering option is difficult to find and unintuitive.	A Company user must be able to search from all Customer’s device information.
2	A Company user must select a Customer or otherwise no devices are found with a given batch date.	Browsing deliveries is unintuitive and time-consuming.	The Company users must be able to browse deliveries from the system.
2, 3, 4, 8	Does not know that the “Search” button opens a window where the search results can be filtered.	The filtering option is hard to find and unintuitive. As the “Search” button is next the search field, users think it is an equivalent to pressing enter.	The buttons in the system must have a corresponding title to the action they execute. The search must have common filtering and sorting options.
4, 8	User has had problems selecting a fault code due to that there are many similar codes. User selects a wrong fault code for the test device as the correct one is not found quickly.	When a code is difficult to find, nothing or a wrong code is selected as the reason. The return reason codes and error codes that can appear on device display are in the same drop-down menu. Therefore, there are duplicates.	Drop-down menus must be searchable. Long drop-down menus must be split by category.
2, 3	Comments that it is misleading that the “Return fault codes” are not in alphabetical order.	Finding an item from the menu is time-consuming.	Drop-down menu items must be in alphabetical order.
7	Customer has sometimes filled the “Customer name” field but not always.	Always filling in the same information to a field is time-consuming.	The system must fill in the information of the logged in user automatically.

1, 6	After each repair batch is done, user lists all the device serial numbers of devices that are sent back for Customer and devices that will be scrapped to separate "Device mass update" views. If devices have not been entered into "Device mass update" view in repair the "Returned devices" and "Scrapped devices" are not shown for the batch. There are multiple scrapped devices in one batch a Customer has sent for repair. But when looking at the batch from "Scrapped devices" there are none.	Changing device states to multiple places in the system is time-consuming and prone to error.	The system must change the devices' state automatically, after sufficient information has been given.
1, 4	Wonders what the function of "Customer delivery ID" field is. Comments about "Customer delivery numbers" that these have just some PO-number. How do I tell these batches apart?	Users cannot tell the deliveries apart by only the IDs and numbers.	The fields in the system must have tips and instructions on their use. The system fields must have a corresponding title to the information is expected to be filled to them. The Company users must be able to browse deliveries from the system.
1, 4	User complains that handling different windows is difficult in the return system. User tries to put the task form and the system side by side on the same screen but then the left side of the system window was missing. User complains that the window contents disappear when moving it or when changing its size. User finds it hard to tell in which system window they are currently working on.	If the user does not have a very large display, the system window fills the whole screen. It is sometimes useful to keep something open by the side. Arranging the system windows is time-consuming. If there are multiple windows open it is hard to differentiate them from their titles.	The UI must be scalable for different display settings and setups. The system windows must be organised so that it easy to browse between them.
4	User noted that there is no English language user manual.	System language can be selected in login. It depends on the selection which language user manual is shown. It would be better if the language could be selected from user settings.	The system must have a user manual available on different languages. The system must have a user settings page where the language can be changed.
1	The system's user manual is aimed for and applicable only for Customers.	When there is no user manual, users may repeat the same errors and some questions are left unasked.	The system must have a user manual that describes the functions also for the Company users.
1, 4	There are two fields with the name "Return fault code". User wonders what information should be entered to the other. Wonders if the date in "Pick-up request form" must be in any specific format.	When the field titles or instructions are not accurate, user spends time wondering what information should be given.	The fields in the system must have tips and instructions on their use. The system fields must have a corresponding title to the information is expected to be filled to them.

1	User comments that it is illogical that when scrapping a device information must be filled to the "Repair fills" side. All information that the Customer fills in should be done to the "Customer fills" side.	There are cases where all users are expected to fill in information to "Customer fills" and "Repair fills" side. This is confusing for the user. This leads to users not filling in required information.	The system must have the knowledge which type of user is currently logged in. The system must show only the relevant fields for the device the user is returning.
4	User wonders which "Save" button should I press when there is information filled on both sides.	Multiple buttons with the same title is confusing for the user.	There should be only one button per page that execute the same action.
1	User clicks the "Save" button and comments that "And then it went somewhere." as the fields go blank.	After the user has taken the time to fill all the fields and then the information disappears without any notice, user is not sure if they were really saved or is all the work lost.	The system must give a confirmation of successful actions.
1, 4	After the devices have been marked for delivery the batch is found from "Export timestamp" drop-down menu. User comments that it is not intuitive and ended up finding it there only by chance based on the "Reference" text they wrote. Complains that this is the tricky part: having to compare timestamps. The correct batch can now be found easier because the test form ask to add a "Reference" text.	The "Export timestamp" and the "Reference" text do not connect in any way if you do not know it.	The system must guide the user through different steps of the process.
1, 2	The "Pick-up request form" is only visible in the system for Customer users. To complete the given Protocol Analysis task, it must be sent in an email.	Company users cannot see the document or some do not know that is there in the system.	All system content must be accessible and available for the Company users.
1	The heading items in the main menu tree are minimised by default. User needs to be guided to find the window for next task.	The main menu is unintuitive and different system windows are hard to find from there.	The system windows must be organised so that it easy to browse between them.
1, 3	Task is to select next Friday for the delivery date. Selects Thursday by mistake.	The date picker is not localised for Finland: the last day of the week in the calendar is Saturday and day names are in English. This causes human error. Selecting a wrong pickup date by mistake creates a recurring error.	The calendar must be localised to the user's preferred language.
1, 2	Not immediately clear for the user that the button with three dots opens the date picker. When clicking the "Batch date" field a (wrong) date "12-3-2021 13:47" appears to it. User has to be guided to press the button with three dots.	The date picker button is unintuitive as three dots does not say anything about calendar.	The buttons in the system must have a corresponding title to the action they execute.
2	User is wondering if the fields with grey background are mandatory to fill.	All drop-down menus in the system a have grey field background. Guessing which fields are mandatory is time-consuming.	The system's user interface must adhere to an agreed style guide.

1	User does not have a barcode reader and types the serial number by hand.	Typing long serial numbers by hand is time-consuming and prone to error.	User must be able to use the system with a phone or a tablet. The system must support using a phone's or tablet's camera to read a barcode to the system.

TABLE 13 User requirements from Protocol Analysis: Reporting and follow-up

User	Thinking aloud	Problem	Requirement
5, 6	In repair we use the CSV export to report for Customers by email what has been done for the devices in the batch. The Customer project manager takes a CSV export from the repair system and informs the Customer of the situation.	Creating and sending the report is time-consuming. Using Excel and email involves multiple steps outside the system.	The system must be able to send notification emails of different events and status changes. The system must offer ready processed reports of selected devices.
4	The system is missing a field where we could write information about the final cause of the fault which could be used in search and see how often it occurs on reported devices. There is a "Repair fault description" field but it is used for different purposes. There is no place in which to record more detailed information about the findings made in the test lab, or what correlates to some fault reported by Customers.	The Company does not get all information of fault causes and loses the opportunity to improve the quality of their products based on that information. Recurring problems can continue without the Company noticing.	The system must support and encourage Company users to add more information about the fault causes. The system must support a process where the fault is investigated. The system must collect fault history and give occurrence reports. Fault codes and descriptions must work as a search criteria in device search.