Communication through boundary objects in distributed agile teams

An integration of user-centered design and agile development

Johan Persson

Supervisors: Johan Blomkvist & Johan Åberg
Opponent: Christopher Palm
Examiner: Arne Jönsson
ISRN: LIU-IDA/KOGVET-A--14/007--SE
Abstract
The use of agile methods continues to increase within software development but the agile processes do not contain the necessary steps to account for the user and realize the usability of the system. It seems it is therefore necessary to integrate the UCD methodology and agile methodology. What this integration should look like is not always apparent as the integration needs to be adapted to the specific context in each organization. Accompanied by a pre-study which identified difficulties with the integration, the current study examines how UCD specialists experience the communication through boundary objects in distributed agile teams. Furthermore the study examines how the understanding of the user is communicated to the developers in the agile teams and the potential of doing so with a design rationale. A case study was therefore performed with interviews of six UCD specialists to try to answers these questions. A content analysis was performed in relation to each research question and identified a number of themes relating to the experiences concerning; (1) communication through boundary objects, (2) how an understanding of the user is communicated, and (3) opinions of using design rationale for this purpose. Some of the conclusions drawn are that boundary objects only aim at communicating the interaction with the system, that they should be viewed as communication aids as they are not stand-alone and making them comprehensive would be even more time consuming. Furthermore, personas are not deemed fitting for communicating the understanding of the user to developers. This should instead be done by allowing developers to participate in user research.
4.2.3 On demand

4.3 What do UCD specialists think about using design rationale to communicate their understanding of the user to developers?

4.3.1 Level and formalization

4.3.2 Time consuming implementation

4.3.3 Relevance

4.3.4 Advantages

5. Discussion

5.1 How do UCD specialists experience the communication through boundary objects with developers?

5.2 How is the UCD specialists’ understanding of the user communicated to developers through boundary objects according to UCD specialists?

5.3 What do UCD specialists think about using design rationale to communicate their understanding of the user to developers?

5.4 Method discussion

6. Conclusions and future research

6.1 Future research

7. References

8. Appendix

Interview template
1. Introduction

Agile development methods are becoming increasingly popular in software engineering as the complexity of systems and their use of context continue to increase. However, it seems that agile development in itself does not provide the necessary insights or opportunities to realize the usability of a system (Hussain, Slany & Holzinger, 2009a). The need therefore exists to integrate agile development with, both the perspective of usability and the methods, which is included in User-Centered Design (UCD). This integration can however be problematic as there is no single unified agile development method. Rather, the term agile development refers to a set of principles for the creation of software. Furthermore, UCD is an umbrella term for a perspective on how usability is accomplished and several methods dedicated to this purpose. As possibilities and obstacles for a successful integration can vary due to differences in organization and project structure, it is necessary to examine and adapt the methods after each individual instance of integration.

The two methodologies also have different perspectives on how and where resources should be allocated (Fox, Sillito & Maurer, 2008) of some of the main concerns with a integration regard who should perform what activities and when to achieve usability (e.g. Ferriera, Noble & Biddle, 2007; Hussain, Slany & Holzinger, 2009b; Sy, 2007). Furthermore, the two methodologies also have different views on documentation. The agile methodology values working software and face-to-face interaction over comprehensive documentation. Conversely, the UCD methodology aims to document the user and the use of the system in as much detail as possible and necessary. This creates the question of what UCD techniques to use and how the result of these should be communicated (Chamberlain, Sharp & Maiden, 2006). The current study and a pre-study were therefore performed in co-operation with a usability consulting company involved in a project employing an agile method. The project was developing an e-healthcare system with the agile method Scrum. Further the project consisted of several services that, in their own right, could be considered individual projects. The complete project was on a national level with several organizations and consulting firms involved. This meant that the members of the agile teams could be distributed on several locations across the country. Due to the size of the project and its structure, the consulting firm in question, felt the need for a study examining the difficulties, obstacles, and possible solutions in relation to a successful integration of UCD and agile development.

The pre-study performed examined the difficulties and obstacles experienced in relation to the integration and identified possible solutions to these based on a created framework. The framework, created from literature relating to the integration of UCD and agile development, consists of four main categories. These are the different roles the UCD perspective can be integrated through, which techniques that are suitable, when in the development process UCD work should be implemented, and how UCD specialists should communicate their work.

Due to the distributed agile teams used in the project, one of the conclusions of the pre-study was the need for further investigation on how appropriate the used artifacts and deliverables were for communicating the UCD work in the distributed agile development. The artifacts and deliverables are therefore viewed through the concept of boundary objects. It also becomes relevant if and how design rationale can be used to aid the UCD specialists’ communication through boundary objects.
1.1 Purpose
The purpose is to examine how UCD specialists experience communication through boundary objects such as shared documentation and deliveries (deliverables). Further the purpose of the study is to examine how the UCD specialists’ understanding of the user is communicated through these boundary objects and what they think of using design rationale as an additional boundary object for this purpose.

1.2 Research questions
In an agile development project:

1. How do UCD specialists experience the communication through boundary objects with developers?

2. How is the UCD specialists’ understanding of the user communicated through boundary objects according to UCD specialists?

3. What do UCD specialists think about using design rationale to communicate their understanding of the user to developers?

1.3 Delimitations
The study is delimited to examine the UCD specialists’ experiences in the specific projects described earlier. This means that the experiences of the developers receiving the documentation and deliverables have been excluded. It is also delimited to only examine any eventual boundary objects created by the UCD specialists and containing information concerning UCD. This examination was also limited to inquiries through interviews as it was not possible to analyze the documentation and deliverables themselves due to restrictions. Although there are more UCD specialists involved in the project than those participating in the study, those UCD specialists were excluded from participating as they had not produced deliverables to developers.
2. Background

The background for the study comes from both the general questions regarding an integration of UCD and agile development processes, and the pre-study performed to examine the difficulties regarding the integration within the specific project.

2.1 Previous Research

The relevant previous research includes User-centered design (UCD), agile methods in general, Scrum, an integration framework, the concept of boundary objects, and design rationale.

2.1.1 User-centered design

UCD is a concept that covers both the methods and the underlying philosophy to create usable systems. UCD strives to create systems with high usability by focusing on the end-user. One drawback of the concept is that it is difficult to agree on a definition as there are many different techniques and methods to achieve usability. The term usability can in itself be defined as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.” (ISO 9241-11, 1998). For the purpose of achieving usability a number of methods have risen. Examples of these are Contextual design (Bayer & Holtzblatt, 1998), Usability engineering (Nielsen, 1993) and Goal-directed design (Cooper et al., 1999). A common trait for these methods is that they all contain three overarching phases; research (requirements gathering and analysis), iterative design and evaluation. The end-user of the product is involved in all these phases to secure the products usability.

In an attempt to provide some clarity to the concept of UCD and to aid projects in its ability to maintaining a user-centered perspective Gulliksen et al. (2004) puts forth 12 key principles of UCD. These are (1) User focus; the importance of the goals of the activity, the work domain and use of context, the users’ goals, tasks and needs as to guide the development. This is usually done by researching the user and summarizing the findings in personas (Adlin & Pruitt, 2006) and scenarios. (2) Active user involvement; representative users should actively participate throughout the entire development process. This could be done by having an actual user participating in key phases throughout the project. It could also, to a limited extent, be done by active work with representations of a user like a persona continuously and throughout the entire project lifecycle (Adlin & Pruitt, 2006). (3) Evolutionary system development; the system’s design and development should be both iterative and incremental. (4) Simple design representation; the representations of the design should be easy to understand by both users and stakeholders. (5) Prototyping; prototypes should be used early and continuously to visualize and evaluate design ideas in cooperation with the user. (6) Evaluate use in context; usability goals and design criteria should determine the direction of development. (7) Explicit and conscious design activities; the development process should include dedicated design activities. (8) A professional attitude; the development team should be multi-disciplinary. (9) Usability champion; UCD specialists should be involved throughout the entire development process. (10) Holistic design; all aspects that will influence the future use of a product should be developed in parallel. (11) Process customization; the implementation of the UCD process must be specified and adapted to each individual organization. (12) A user-centered attitude should always be established; everyone involved in the project must be informed and dedicated to the importance of usability and how it can be achieved. According to Gulliksen et al. (2004) these 12 key principles covers the analysis, design, evaluation, construction and implementation phases in the
development process to create usable systems. Furthermore these principles enable the development, communication and evaluation of the UCD process.

In relation to the first two of Gulliksen’s principles and for the purpose of representation, personas are commonly used. A persona is a figurative user devised from the user research to aid the design and development process. According to Adlin & Pruitt (2006) one of the most difficult and demanding tasks in relation to personas is the transition from creating a persona to using one. They point out that for personas to be useful to its fullest extent, one must progressively present information in small and easily interpretable chunks constructed after the intended receiver.

2.1.2 Agile software development
A classical software development process according to the waterfall model usually contains four basic phases; research, design, development, and evaluation. These four phases’ takes place separate and sequentially to each other and the focus lies on comprehensive planning and a clear and structured process (Blomkvist, 2005). Between each of the phases a delivery takes place which often contains extensive and detailed text documents. One of the problems with the waterfall model is that it’s static while the reality of the outside world is dynamic with often changing technical conditions and requirements. One additional problem is the delayed feedback which only occurs once there is a functional version of the system at the end of the development process. The CHAOS report (Standish group, 1995) reports that only 16 percent of software development projects are delivered on time and according to budget. Further they report that the three most common problems within software development is (1) a lack of user input, (2) incomplete requirements and specifications and (3) changing requirements and specifications. As a result of these problems and to cope with a faster changing world, largely due to the rise of the internet, agile software development began to find traction (Abrahamsson et al., 2003). Most of the agile methods grew up during the 90s but were then called “lightweight” methods. These methods originate from early project methods in the 50s and 70s like Iterative and Incremental design (IID) and Evolutionary project management (Evo) respectively (Sliger & Broderick, 2008). There isn’t a unified agile method but the different methods contain several common elements like iterative and incremental development, adaptive and flexible planning, and a focus on communication and people (Larman & Basili, 2003). Every iteration within agile development can be seen as a miniature development process consisting of the four previous mentioned phases.

**Agile Manifesto**
As there is no single unified agile development method, 17 experts within the “lightweight” methods gathered in 2001 to discuss the similarities and common elements of their methods. As a result the agile manifesto was created along with the term “agile methods” being coined. The manifesto is consisting of four main values accompanied by 12 principles to express the underlying philosophy of the agile methods (Agile Manifesto, 2001).
Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck  
Mike Beedle  
Arie van Bennekum  
Alistair Cockburn  
Ward Cunningham  
James Grenning  
Jim Highsmith  
Andrew Hunt  
Ron Jeffries  
Jon Kern  
Robert C. Martin  
Steve Mellor  
Ken Schwaber  
Jeff Sutherland  
Dave Thomas

The accompanied twelve principles are as follows (Agile Manifesto, 2001):

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity — the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Scrum
Scrum is an agile development method created by Ken Schwaber and Jeff Sutherland. Scrum is a result of coping with the complexity accompanied by developing software (Schwaber, 2004) and does this by focusing on a variation of project management values (Larman, 2003). By placing focus on project management Scrum is considered suitable to be implemented with other agile methods (e.g.
Extreme Programming) that give a more detailed work practices such as coding standards. Scrum uses an empirical process control instead of a defined process control to cope with the complexity of software development. The empirical process control consists of three cornerstones; Visibility, Inspection, and Adaptation. (1) Visibility means that the aspects of the process that affects the outcome must be visible for those who control the process. (2) Inspection means that the various aspects of the process must be inspected so that unwanted variation can be discovered. (3) Adaptation means that if the inspection results in finding unacceptable variations that can influence the product in a negative way, the process must be able to be adapted to solve the variations. A basic element in Scrum is the iterative and incremental process which is called a sprint. A sprint normally lasts between 7-30 days. Every day during a sprint an inspection is held in the form of a team meeting. At the end of every sprint a functional increment of the system is delivered to the stakeholders who inspect the functionality and any adaptations to the project are made (Schwaber, 2004).

There are three types of roles within Scrum. These are the product owner, the scrum master and the development team. The product owner is responsible that all the stakeholders interest are represented and for the product and its functionality. The product owner is also responsible for creating and maintaining the product backlog which is a list of requirements and planned functionality represented in the form of user stories. The maintenance of the backlog includes prioritizing the requirements and functions so that the most important ones can be implemented first and be built upon (Schwaber, 2004). The scrum master is responsible for the scrum process and to teach Scrum to everyone involved. Further the scrum master is responsible for the implementation of Scrum in the organization so that it fits and can deliver the expected advantages. The scrum master’s responsibilities also includes making sure that the team follows the practices and rules of Scrum (Schwaber, 2004). The team’s responsibility is to develop the functionality of the system. The team should be self-governing, self-organizing and consist of cross-functional members. The team is also responsible for turning a part of the product backlog into a functional increment within a sprint and for the product as a whole.

All the work is carried out in sprints which start with a planning meeting. During the planning meeting the product owner cooperates with the team to determine what will be done in the next sprint. First the product owner presents the highest priority backlog items. Second the team questions the content and its purpose in order to acquire the understanding needed to decide how much can be done in the sprint. By giving story point to each user story the team arrives at how much from the backlog that is possible to deliver as a functional increment (Schwaber, 2004). After consensus is reached between the team and the product owner the team carries on by breaking down the user stories into tasks which are then added to the sprint backlog. The work progress can then be monitored by following a burn down chart. Every day during the sprint a stand up meeting (or Daily Scrum) is held which lasts about 15 minutes. The meeting aims at inspecting the team members work, synchronize the work and plan for upcoming tasks (Schwaber, 2004). During the meeting every member answers three questions:

- What did I do yesterday?
- What will I do to tomorrow?
- What hinders me from accomplishing it?
At the end of each sprint a sprint review meeting is held where the team presents the developed increment from the previous sprint to the product owner and the stakeholders. During the meeting the functionality of the increment is examined to discuss what the team should do next. After the review meeting and before the next planning meeting a sprint retrospective is held with the scrum master and the team. The purpose of which is to evaluate the previous sprint and to apply any changes if necessary to the project (Schwaber, 2004)

2.1.3 Integration of UCD and Agile methods
There are a couple of general problems concerning the integration of UCD and agile methods. Agile methods are a clear improvement from the classical waterfall model in relation to shorter feedback loops and earlier evaluations. Feedback occurs at the end of each iteration through the review meetings where the customer can give their opinions about the delivered increment. The distinction between end-user and customer is not always made within the agile methods which means that usability is not necessarily realized (Hussain et al., 2009a). The problem with the integration can here be related to a cultural difference between the people within these two fields and what they see as usability and how to achieve it. Blomkvist (2005) argue that the integration of UCD and agile methods can be seen from an abstract and concrete perspective. The abstract perspective relate to how the underlying philosophy of UCD and the view of how usability is achieved can be integrated with the agile principles. The concrete perspective on the other hand relate to how specific processes or techniques can be integrated with each other.

As a result of the previous mentioned agile principles, agile methods oppose that extensive research and analysis precede development. This is in direct opposition to how UCD strives to create an understanding of the user by gathering requirements through user research. Although the iterative process in agile methods do create an opportunity to implement UCD techniques in a timely fashion for creating a usable product. One of the main problems for an integration of the methods is how and when UCD techniques should be implemented as well as what techniques that are appropriate (Chamberlain et al., 2006; Ferreira et al., 2007; Hussain et al., 2009b). Blomkvist (2005) identifies three different approaches for integration of UCD and agile methods. The first two of these is either integration of UCD techniques and methods into agile methods or agile methods into UCD methods respectively. The third and preferred approach according to Blomkvist is a balanced integration, which would result in a better adaption of the underlying values of both methods.

A framework has been created out of the previous research in the field and is presented below. The framework includes how UCD as a whole can be integrated with different roles, techniques, timing, and communication and collaboration

Roles
According to Fox et al. (2008) one of the main differences between the different approaches of integrating UCD and agile methods is by whom the UCD work is performed. They identify three different approaches which they call the generalist, specialist and hybrid approach.

In the generalist approach the UCD work is performed by some or all developers who have some form of informal UCD training. The UCD work in the generalist approach means that only a small amount of time is set aside for its execution. Furthermore the execution is less formal and evaluation tends to be performed with members of the team or the customer and towards heuristic principles (Fox et al., 2008). One of the drawbacks with the generalist approach is that the UCD techniques risk
being inadequately executed and with not enough time. Further, Ferreira et al. (2007) reports that projects with informal UCD practices also risk usability issues to go unsolved in already developed increments of the system. As a UCD specialist is missing the generalist approach also violates the agile principle of cross-functional teams (Blomkvist, 2005).

The specialist approach means that a member of the development team holds a formal UCD training (Fox et al., 2008). This member performs user research separate from the rest of the team and before development starts to gather requirements. During this time the specialist also sketches design solutions in the form of wireframes and creates lo-fi prototypes which are then evaluated with users. This work is usually done within a four to six week timeframe (Fox et al., 2008). After the development start the UCD specialist continuously acts as a bridge between the user and the team, by performing new user research and evaluations. One of the drawbacks with the specialist approach is that user knowledge won’t be as distributed among the team.

The hybrid approach is when a team member has experience from both programming and formal UCD training. Fox et al. (2008) found that this approach was not as widely used as the other two, where only two of the ten projects in their study used it. One of the main differences to the specialist approach was the occurrence of a UCD team containing several UCD specialists. The team member with experience from both fields then acted as a bridge between the two groups.

**Techniques**

Due to the relatively fast iterative processes, the view on documentation in agile development and the individual differences between projects, it is not certain that all UCD techniques fit in. Hussain et al. (2009b) performed a survey with 92 developers and UCD specialist to identify the most common used UCD techniques in agile development. The results showed that the most common techniques was lo-fi prototypes followed by conceptual designs, observational studies, expert usability evaluations, field studies, personas, and rapid iterative testing.

Chamberlain et al. (2006) account for the successful use of personas and scenarios in agile development. Sy (2007) describes that depending on whether it is a completely new product or a new version of an existing product they would create personas and scenarios or summarize previous user research respectively. Bayer et al. (2004) suggests with their method Rapid contextual design that user research and contextual inquiries should be used to create user stories as these can be considered to be relatively detailed definitions of a system’s functions. These should in turn be used to create conceptual designs. Patton (2002) presents the use of user stories as criteria for acceptance tests. According to Brown et al. (2008) user stories can be used to aid communication between developers and UCD specialists. User stories does this by making problems apparent, defining delimitations, articulating requirements, and connecting designs to user groups and goals. Furthermore, Brown et al. (2008) also analyzed the use of sketches in relation to communication. The result show that sketches are well suited to focus attention, grounding communication, recording design decisions, and to explore relationships within a design.

As mentioned lo-fi prototyping are the most common UCD technique in agile development according to Hussain et al. (2009b). Lo-fi prototypes can be anything from sketches on a whiteboard (Brown et al., 2008), an arrangement of post-it notes and wireframes (Patton, 2002) to storyboards (Chamberlain et al., 2006), sketches with pen and paper (Bayer et al., 2004), and power point prototypes (Ferreira et al., 2007). Lo-fi prototypes can also be used with a formative or evaluative
purpose. Fox et al. (2008) presents examples where lo-fi testing is used as a part of the user research to gather requirements and to extract new functionality in an early stage. They also present examples where lo-fi testing occurs so designs are evaluated and verified before handed over to the development team. Meszaros & Aston (2006) also puts forth testing of lo-fi prototypes as means of getting early feedback on requirements and planned functionality. According to Ferreira et al. (2007) the use of lo-fi prototypes and the iterative process of agile development open up the possibilities of rapid iterative testing and implementation of its results. Rapid iterative testing is also one of the most common used UCD techniques (Hussain et al, 2009b). Although with some advantages, these do not have to be executed with lo-fi prototypes. A couple of examples are presented where functional increments of the system are used for verification and evaluation. In one example a functional increment is used for verification and evaluation of the functions in the system and there after approved or revised (Fox et al., 2008). Another example is from Miller (2005) and Sy (2007) who describe the use of several functional increments to test and evaluate larger work flows with actual end users.

**Timing**

For a successful integration, the previous mentioned UCD techniques needs to be adapted and implemented in a timely fashion. In accordance with the agile manifesto methods like Scrum oppose the basic principles of UCD, that thorough user research and requirements gathering precede development. Conversely, some sort of initial work seems to be necessary to create a vision of what needs to be built and consequently steer the development (Meszaros & Aston, 2006; Chamberlain et al., 2006; Nodder & Nielsen, 2008). For a successful integration both UCD and agile methods need to compromise on how much work and time precedes development. Several sources in the literature describes how a timely integration can be accomplished with the presence of a pre-sprint and by working in parallel tracks (Bayer et al., 2004; Miller, 2005; Sy, 2007; Fox et al., 2008; Nodder & Nielsen, 2008; Adikari et al., 2009; da Silva et al., 2011). An adapted work process model of Sy (2007) and da Silva et al. (2011) is presented below showing the different stages related to the pre-sprint and parallel tracks (see Figure 1).

During the pre-sprint (sprint 0) user research and contextual inquiries are performed to create user stories and paper prototypes (da Silva et al., 2011). A similar work process is presented by Adikari et al. (2009) which they call Little design up front. In Little design up front user research and lo-fi testing is conducted prior to development to gather just enough information to be able to support development in the sprints later on. According to Fox et al. (2008) one commonality for the different projects under study was the use of a pre-sprint which lasted four to six weeks. During this time initial user research was conducted while also creating lo-fi prototypes and testing them. As previously mentioned, Miller (2005) and Sy (2007) either performed user research or summarized previous research, depending on the type of product to extract design goals during the pre-sprint. The result of the pre-sprint was usability requirements in the form of user stories and lo-fi prototypes which was handed off for implementation. The UCD specialist then started working in a parallel track with the development track (see Figure 1).

In the parallel track and during sprint 1 the UCD work continues by creating user stories, designs and iteratively testing these with users for the upcoming sprint (Bayer et al., 2004; Miller, 2005; Sy, 2007; Fox et al., 2008). It is also important for the UCD specialist to support the development team in the current sprint with daily communications (Miller, 2005; Sy, 2007; Ferreira et al., 2007). According to
Sy (2007) one prerequisite to the parallel tracks is the ability to break down a larger design into design chunks. This is done in a similar fashion to how agile methods break down larger systems into functional increments in each sprint.

![Work process model with pre-sprint and parallel tracks. Adapted from Sy (2007) and da Silva et al. (2011).](image)

During the second sprint the UCD work continues in a similar fashion to sprint 1 with the creation of user stories, design and iterative testing for the upcoming sprint (sprint 3). At the same time user research is conducted for sprint 4. As in the previous sprint the UCD specialist support the development team in the current sprint but in addition performs usability evaluation and verification of the implemented design from sprint 1 (Miller, 2005; Sy, 2007; da Silva et al., 2011). Any feedback of the results from the evaluation is implemented in the same or next sprint, depending on size. The UCD work then continues in a similar fashion in the next sprint by evaluating implemented designs from sprint n-1, supporting the current sprint, creating user stories and designs for sprint n+1, and performing user research for sprint n+2 (see Figure 1).

**Communication and Collaboration**

The agile manifesto values individuals and interactions over processes and tools. A systematic literature review (da Silva et al., 2011) indicates that agile projects collaborate closer and has better communication between developers and UCD specialists. Raison & Schmidt (2013) identified co-location of developers and UCD specialists as a contributing factor for successful projects. According to Chamberlain et al. (2006) UCD specialists and developers has to be willing to collaborate on a daily basis and claims that all members need to be participating in the key phases so not to create communication problems later on. Patton (2002), Bayer et al. (2004), and Ungar (2008) all presents examples of how developers participate in early phases before development with some form of user research and requirements analysis. Although Patton’s example is lacking a UCD specialist, developers collaborated with domain experts and stakeholders in a workshop to create an idea of the user and a prototype as the end result. This was done to distribute the knowledge of the user
and to use the created artifacts as a communication aid throughout the project. Rapid contextual design (Bayer et al., 2004) gives an example where UCD specialists invite the developers for a couple of days to collaborate and decide what problems the projects should address and try to solve. Ungar (2008) presents the use of a design studio to improve collaboration and the understanding for the design process. The design studio takes place one day before development and was attended by UCD specialists, developers, and stakeholders.

Furthermore Chamberlain et al. (2006) claims that there has to be a mechanism or role to ensure that usability requirements are considered. Out of the projects in their study, the project with a design lead experienced the least amount of problems with collaboration. The design lead’s responsibility was to extract requirements with the leads from the other disciplines. Budwig et al (2009) presents a similar solutions where the design team had their own product owner whose responsibilities included leading the UCD work, act as a bridge to the regular product owner and managing usability requirements and deliverables. According to Budwig et al. this ensured that usability issues were discussed in the planning for each iteration.

As previously mentioned agile methods does not put high value on comprehensive documentation which means that UCD specialists have to find other artefacts to communicate through than extensive design specifications. One of the most common ways to communicate usability requirements seems to be through prototypes (Bayer et al., 2004; Meszaros & Aston, 2006; Ferreira et al., 2007). According to Fox et al. (2008) the work of UCD specialists is to prepare for the next iteration planning. Meszaros & Aston (2006) presents an example where a lo-fi prototype was used to communicate user requirements during the planning phase. The prototype was later turned into a UI story board where user stories were connected to it. The prototype was further integrated in the iteration planning by aiding in the prioritization of user stories for the next iteration. Further, the authors suggest that the story board enabled the team members’ communication by creating a shared understanding and vision.

2.1.4 Boundary Objects

Communication and the grounding of information in distributed agile teams become more difficult due to the lack of grounding factors present to the teams (Modi, Abbott, & Counsell, 2013). As coordination of information depends on knowing what knowledge is necessary to share (Convertino et al. 2009), having awareness of team and task activities thus becomes vital. This means that the importance of some boundary objects, for which the team members can communicate through, grows as they should also include a mediation of the underlying activities for the boundary objects. Furthermore these boundary objects become important as a reference for other communication in distributed agile teams as they allow actors from different communities to negotiate meaning and create a shared understanding (Modi et al., 2013).

The term boundary object was first coined by Star & Griesemer (Star & Griesemer, 1989) and is used to describe a variety of objects that inhabit different worlds of knowledge. Boundary objects serves as a medium for actors from different cultural, social and scientific backgrounds. According to Star & Griesemer, boundary objects is “an analytic concept of [those] scientific objects which both inhabit several intersecting social worlds...” (Star & Griesemer, 1989, p. 393). Further they describe boundary objects as both flexible enough to adapt to individual actors and the constraints of several actors.
employing them. But they are also rigid enough to maintain a common identity across actors and backgrounds.

Star & Griesemer presents a few examples of different types of boundary objects (which are not meant to be exhaustive). The different types are repositories, ideal type, and standardized forms. An example of a repository is a library where actors from different worlds can use or borrow information for their own purpose. An ideal type is exemplified by objects such as diagrams or maps. They do not describe the detail of one specific thing and are abstracted from all domains by being vague. Its purpose is to serve as a symbol for communication and collaboration. Standardized forms are objects used for common communication across distributed groups and is used to eliminate local variation of information.

2.1.5 Design Rationale
Design rationale is used to make the reasoning process behind a design and its elements apparent through documentation. According to Moran & Carroll (1996), the first to propose the use of explicit design rationale was Horst Rittel (Rittel & Weber, 1973). Rittel viewed design as a number of negotiations and deliberations for what he called “wicked problems”. These problems are design problems which lack a definite answer due to the many subjective criteria for evaluation. When solving design problems one must deal with uncertainty and conflict, where one solution to a design element may interact with others and the consequences create conflict for the design goals. The recognition of design problems complexity and the subjective perspective of evaluation led to the need of an explicit rationale behind the designs (Moran & Carroll, 1996).

There is no agreed upon definition of design rationale and the term is used in many different ways across different fields and in literature. Shipman & McCall (1997) presents three different but overlapping perspectives of design rationale in their article. The first of these perspectives is the argumentation perspective which aims at recording the reasoning that the designers use for framing and solving the design problem. This perspective is meant to aid the designers and the design process by explication of arguments and framework. This will in turn lead the designers to reflect about the rationale and its deficits and therefore improve their argumentation.

The second is the documentation perspective. This includes documenting what decisions are made, when they are made, why they are made, and by whom (Shipman & McCall, 1997). The documentation perspective records less information in the design rationale than in the other perspectives. This is due to one of the motivation for the documentation perspective which is for people outside the project to understand and supervise it. And so it is only necessary to record the result of the reasoning and its direct explanation. According to Shipman & McCall (1997) the value of the design rationale in the documentation perspective may be very low if design decisions are not explained or justified clearly, even though the documentation perspective doesn’t strive to influence decisions itself. Moran & Carroll (1996) describes the documentation perspective in three different ways, from a specific to a general sense. In the first and specific sense it refers to just those parts that give reasons for the design. The second sense is exemplified by design notebooks which document the different stages of the design process and show how the final design was arrived at. In the third and general sense, the entirety of the documentation in a design project becomes the design rationale as it shows the organizational, social, cultural and political influences on the final design (Moran & Carroll, 1996).
The third is the communication perspective. In this perspective, design rationale contains the natural conversations between group members in a project. The aim of design rationale in a communication perspective is to enable retrospection of communication during the design process. This means capturing ideas during the design process and not giving rise to them. According to Shipman & McCall (1997) the communication perspective does not have a separate goal of its own but instead overlap with both the argumentation and documentation perspectives. The communication perspective also tends to be less structured and contain irrelevant information in relation to the design.

One of the difficulties with design rationale is the capture of it. For design rationale to be useful it must first be captured. For capturing design rationale it must first be represented. The representation of design rationale can be informal or formal. The consequence of whether the representation of design rationale is formal or informal extends to both the capture and use of it. By formalizing the capture of design rationale, the terms of representation becomes constrained and can hinder the design process (Moran & Carroll, 1996). On the other hand, the retrieval and use of the rationale can be aided by computer systems. Conversely, by informal capture a wider array of rationales can be captured but the retrieval and use of it can later become strenuous. The capture, management, and retrieval of design rationale can be heavily time consuming for both the formal and informal approach. It is therefore necessary to integrate the tools by which design rationale is managed in the daily activities and artifacts already in use (Moran & Carroll, 1996).

There are several different suggested systems in the argumentation perspective. A few of these are the issue-based information system (IBIS), decision representation language (DRL), and design space analysis. Lee (1997) has through a survey of existing design rationale systems identified a generic structure. According to Lee the structure of design rationale consists of three layers; the design intent layer, decision layer, and design artifact layer. The design intent layer is the highest level of design rationale. It is here the information driving the decisions in the design process is represented in the form of intents, strategies, goals, and requirements. The decision layer contains the information (rationale) concerning the decisions in the design process. It consists of five sub layers which show the structure of decision process. The five sub layers are issue, argument, alternative, evaluation, criteria. The issue layer consists of individual issues and their relations (generates, depends on, replaces). The argument layer expresses the arguments underlying the decisions and their relations (supports, refutes, qualifies). The alternative layers shows individual alternatives and their relations (component of, incompatible, specializes). The evaluation layer expresses the evaluation measure and relation between alternatives and the measures. The criteria layer contains the criteria for evaluating the alternatives and can for example be formulated from information in the design intent layer. The design artifact layer shows relation between design components and can also link these to the underlying rationale (decision layer).

Falessi et al. (2013) has identified further difficulties with the implementation of design rationale such as information predictability and bad timing. The first is concerned with predicting what information is necessary for the consumer of the rationale, and the second indicates that individuals involved in design decisions are usually occupied with more essential tasks leaving the rationale to eventually be dismissed. The timing becomes even more essential when the implementation occurs within the rapid pace of agile development. The question is therefore, what room is there for a design rationale in the agile methods that also put low value on comprehensive documentation?
Sauer (2003) argues that agile development do not render documentation as design rationale obsolete. He puts forth an event-based design rationale with semi-automatic capturing. By arguing that a project is a set of events that leave a project trace, that project trace can later be used to evaluate new decisions by comparing to previous events. It does this by using prototype arguments and design rationales (i.e. argument prototypes and DR prototypes). Each argument and rationale gets a weighted value which can later be used to calculate if an event satisfies criteria.

2.2 Pre-study
The purpose of the pre-study was to examine what obstacles existed for the integration of UCD and agile development, and what these obstacles looked like. Furthermore, the purpose was to identify possible solutions to these problems. The research question for the pre-study was therefore: What are the problems in relation to an integration of UCD and agile development, what do these look like and what possible solutions exist?

2.2.1 Method
The pre-study took the form of an explorative case study. Case study as a research method is used to investigate a current phenomenon within its real-life context and is most suitable when the boundaries between the phenomenon and the context are unclear (Yin, 2007). In the pre-study, integration of UCD and agile development is considered to be the phenomenon under investigation and its context the development project that the consulting firm is involved in. Possible sources within case studies are documentation, archives, interviews, observations, and physical artifacts. An important concept in case studies is triangulation, which means the verification of information through multiple sources. In addition to the previous mentioned sources triangulation can be accomplished by methods, researchers and different perspectives in literature (Yin, 2007).

Interviews were conducted with four participants with various roles in the project but all from the same consulting firm. Participant 1 was the consulting firm’s project manager for the project and had a background within UCD. Participant 2 was a UCD specialist in the project and was for a time also responsible for the planning of the project. Participant 3 was one of the consulting firm’s own front-end developer and was involved with two of the sub-projects (or services). Participant 4 was both front-end developer and UCD specialist but took onl only the official role of one at a time. This meant that participant 4 was acting as a front-end developer in one of the sub-projects and a UCD specialist in another.

The primary source of information was interviews. The interviews aimed at exploring the current structure of the project and the participants’ experiences with obstacles to a successful integration of the two methodologies. The interviews were semi-structured and lasted about one hour per participant. An excerpt of some questions is presented below:

- What part of the project are you involved in?
- What tasks do/did you participate in?
- What is your current understanding of the project process? (e.g. relationship between requirements, design, development, testing)
- What deliverables do you have?
- What does the deliverance look like?
- How do you communicate within the project?
- How involved are you in the creation of backlogs?
• What does your everyday work look like?
• If and how do you keep up the daily communication with colleagues in the project? What are the biggest problems?
• What do you experience to be the biggest obstacles for an integration of UCD and agile development?

Each interview was summarized by categorizing and thematising the problem areas in the answers. They were then compiled with each other to identify patterns between the participants. Identification of general categories and problem areas could then be made. Additional analysis was then performed in relation to the framework, which had been created from the relevant literature, to identify possible solutions to the identified problems.

2.2.2 Results
Themes and problem areas that have been identified and categorized are presented below. These are planning, structure and, communication. There is a natural occurring overlap between the problems and categories. The categorization has been made according to what is seen as the underlying cause or by the possibly appropriate solution.

Planning
The interview has shown a couple of problem areas in relation to planning. One of the bigger problems was the lack of time allocation for a pre-sprint where early UCD work could be performed to clearly specify requirements. According to participant 2 (P2), the project planning is in too large extent based on the development and that the UCD work simply has to adapt. Furthermore, P2 claims that the project planning does not consider from where the requirements should be established. The result is that developers only get deliverables, grounded in requirements, when there is time. Consequently, the UCD specialist does not succeed in compiling the material properly to communicate a comprehensive and clear picture. In that sense there is a shift of responsibility to the developers to personally go through artifacts like personas and scenarios to acquire a comprehensive and clear picture. P2 stresses the importance of a pre-sprint to accomplish usability in a product and that work outside of development also shows in a product. Participant 3 (P3) said that the lack of a pre-sprint means that UCD work to extract usability requirements takes place in the same or one sprint behind development. Consequently, questions concerning requirements always take place on the spot and that new requirements keep coming in for the current sprint. P3 claim that this result in a time shortage for UCD specialist in the production phase. P3 attribute these problems to deficient planning and the lack of prioritizing. In P3’s experience there is no big picture planning and requirements should be able to be anticipated if there was one. Furthermore, P3 thinks that it is important to have a dedicated UCD specialist on the team.

The second problem area related to planning is the iteration cycles and the synchronization between development and UCD work. According to Participant 1 (P1), the UCD work is not performed after the developments sprint planning, instead it is planned and takes place in a separate track. P1 claims that this is because the UCD work is too complicated to break down so it fits within the sprints. P3 expresses that this causes problems as deadlines varies too much between teams and results in unorganized intake of requirements in the development sprints. P3 also expresses the appearance of a disjointed UCD team which causes them to getting behind in the sprints. Participant 4 (P4) claims that if the UCD team collaborated closer together by identifying and prioritizing tasks it would possibly enable breaking down the UCD work into the sprints.
Structure
The category structure is closely related to planning but the problems presented below are considered to be more of natural consequences of the type of project rather than a lack of planning. P2 explains that the optimal work flow one wishes to have is not always possible due to the customer’s order, time and cost. The work process is therefore based and adapted to the order. P2 also feels that the fast development pace in Scrum results in that developers do not have enough time to aid the UCD specialists about any eventual technical constraints and solutions. P3 and P4 both experience problems with not being co-located with the other developers. Working in distributed teams can be considered a structural problem with implications on communication. It is therefore describe closer in the next sub-chapter. P4 also expresses the fall out of sprint retrospectives as a problem.

Communication
The identified problems in relation to communication varied from general down to specific problems. P2 experience a good communication with their own front-end developers but not with the back-end developers which are not co-located. Although P2 points out that the communication with the front-end could also have been better. Partly, the problem seems to be that the UCD specialists have a full schedule with meetings and that they’re not working full time on the project. P2 claims that if they had been working full time the communication would probably been better. They could then been co-located and seen how the developers work. In spite of this, P2 means that attending the daily stand-ups did not contribute to the communication as developers and UCD specialist speak different “languages”. P3 and P4 which both have the role of front-end developers reported the same experience and did therefore act as a bridge between the UCD specialists and developers. P3 was still experiencing the communication as problematic and suggest that there should be a UCD lead with the technical experience required to attend the daily stand-ups. P2 expresses a similar opinion with the need of an owner for the usability requirements. According to P2 this is required to ensure a focus on usability during the planning of each sprint. P4 states that communication and discussions during the delivery meetings are also insufficient as participants are not adequately informed about each other’s areas. P4 proposes a separate delivery meeting so that usability can be properly discussed.

P2 stresses the importance of the need of developers’ empathy for UCD specialists and their work but also UCD specialists’ empathy for developers work. P2 recon this could be accomplished by co-location or alternatively a design workshop. P3 and P4 also raise the problem of not being co-located with the developer team. P3 claims this creates a riff between the members and that they miss important and spontaneous discussions. P4 means that communication becomes verbose and can take up to much time. Furthermore, P4 feels that design specifications should include clearer motivations in relation to design elements and that UCD specialists need to attend the daily stand-ups to prevent small issues becoming big issues.

2.2.3 Discussion
This chapter will firstly discuss the results in relation to the literature and later the method of the study.

Results
One of the main categories identified from the interviews and in relation to an integration of the two methodologies was planning. Within the category a lack of a pre-sprint was seen as a contributing
factor to the experienced problems. The natural consequence of this is that the requirements needed for a current sprint is missing at the time of the planning meeting and UCD specialists will therefore struggle to catch up. According to the framework created from the literature a pre-sprint is necessary for the timely and effective implementation of UCD work in relation to the development process (Sy, 2007; Adikari et al., 2009). Furthermore, it seems that the lack of a pre-sprint results in unstructured intake of requirements during the sprints. One expressed opinion contributed this to insufficient planning and that a large amount of requirements should be able to foresee with an organized vision. The framework, in accordance with this, shows the importance of a pre-sprint to generate a clear vision which can steer the development in an early stage (Meszaros & Aston, 2006; Chamberlain et al., 2006; Nodder & Nielsen, 2008). Chamberlain et al. (2006) claims that an integration of the two methods have to take place within a project management that is not too bureaucratic or predetermined. Even though the agile methods are against pre-work there has to be a compromise for a successful integration (Fox et al., 2008).

The iteration cycles and the synchronization between UCD and development was another of the main factors to the problems identified in relation to planning. The UCD work takes place in a different track for that of the development which results in difference in deadlines. Consequently, requirements for a current sprint could arrive at different and unfortunate times. According to Blomkvist (2005), a prerequisite for a successful integration is that it is balanced between UCD and agile development. This means that UCD work does not only have to be integrated into agile development, agile development has to also be integrated in UCD. A possible solution for this problem is presented in the framework in the form of parallel tracks. A prerequisite for the parallel tracks is that the UCD work is performed one or two sprint ahead of development by planning a pre-sprint. The UCD work during the later regular sprint aims at preparing for the upcoming sprints by performing user research and testing prototypes. The agile iterative and incremental process is integrated in the UCD work, which happens in parallel tracks, by initially breaking down a larger design into smaller design chunks. These can later be prioritized and addressed separately in the consecutive sprints (Sy, 2007). Furthermore, a compilation of user research and usability evaluation enables a timely and sustainable work process within the sprints (Miller, 2005; Sy, 2007).

The problems identified in the category of structure are foremost related to the implementation of Scrum. Two of the cornerstones for the empirical process control, that Scrum is based on, are inspection and adaption. The absence of the sprint retrospectives may have a strong influence of the outcome of the project as this meeting enables inspection and adaption. The agile manifesto values individuals and interaction but the interaction is undoubtedly and unnecessary complicated by a distributed development team. Co-location of UCD specialists and developers becomes more important as Raison & Schmidt (2013) identified it as a contributing factor for successful projects. It is possible there are measures for which these problems can be solved but none that was explicitly expressed in the current literature.

One of the larger problems in relation to communication was that UCD specialists and developers speak different “languages”. As a solution to the problem the consulting firm used their own front-end developers. These developers then acted as a bridge between the UCD specialists and back-end developers in a similar manner to what Fox et al. (2008) suggest in their hybrid approach. The difference being that the hybrid role neither acts as a UCD specialist or developer directly. Although the consulting firm used its own front-end developers problems was still experienced by the
participants. Two of the participants pointed to the demand of a UCD lead that would own the usability requirements. This suggestion coincides with the framework. In Chamberlain et al. (2006) the project with the least experienced collaboration problems used a design lead. Budwig et al. (2009) also presents a similar solution where the UCD team had their own product owner which was responsible for usability requirements and acting as a bridge towards the development team. According to Budwig et al. this improved the collaboration and ensured a focus on usability during the planning meetings.

One further problem in relation to communication was the importance of empathy between the developers and the UCD specialists. The problem that is expressed is the lack of understanding about the UCD process and what the work means from a developers perspective. One participant suggested a design workshop. This solution is in accordance with the framework, which points out that all members have to be participating in some key phases of the process (Chamberlain et al., 2006). The framework includes a couple of different measures to raise awareness among developers and stakeholders about the design process and to improve collaboration. This can either be done by holding a workshop (Patton, 2002), where all members collaborate to create an understanding for the user and their goals. Or through a design studio (Ungar, 2008) where volunteers came with pre-worked design solutions that later was went through to arrive at a conceptual design.

While two participants viewed attending the daily stand-ups as problematic, one participant thought it was necessary. The motivation for this being that UCD specialist had to be present to prevent small issues concerning the design specification to become larger issues. According to the framework daily communication is necessary to support development (Miller, 2005; Sy, 2007; Ferreira et al., 2007). Miller (2005) gives an example of where the design specifications were not “just thrown over a wall” but was successively handed over by attending the daily stand-ups. This allowed the developers to easier understand the designs and it underlying motives. Furthermore, some sources in the literature suggest using other artifacts, such as prototypes, sketches and user stories, rather than design specifications to communicate with the developers. Bayer et al. (2004) suggest the use of user stories as these are relatively detailed definitions of the system’s functionality. Further Brown et al. (2008) suggest the use of user stories to communicate requirements, delimitations, and to connect users and their goals to a specific design. They also suggest that prototypes are fitting for grounding discussions between developers and UCD specialists, and to record design decisions and issues. Lastly Meszaros & Aston (2006) give an example of the use of a prototype to communicate usability requirements. They also show how the prototype while connected to user stories served as a foundation for the prioritization for upcoming sprints.

Method

The case study was performed with interviews and only a small part of documentation as information sources. It is therefore possible that the study’s validity can have been affected. The documentation collected consisted of planned and actual implementations to the project and could therefore only partly be used for triangulation. The documentation was therefore mostly used for preparation of the interviews.

One disadvantage to the study and to the literature as a whole is the reliance on anecdotal empiricism through interviews. Interview emphasizes undoubtedly important information on how integration appears, although observations would give the possibility to examine any contingent
aspects of integration. Observations would also enable triangulation and should be used as a complement to interviews to strengthen the validity. The absence of observations in the study was due to the time and access constraints and could therefore have influenced the study’s validity.

One further aspect that can have affected the study negatively is that the literature was collected and worked through at the same period as interview was conducted. Due to the fast start up of the study, the interviews were conducted before the bigger part of the current literature had been read. This could have had a positive effect as interview questions was created and asked without any prejudice or expectations of specific answers. The negative side is the risk that important questions were missing which would have led to aspects being overlooked. However, the categorization of problem areas from the interviews largely coincides with those in the literature, which therefore supports the study’s validity.

The interviews were performed early on but with some space in between. The interviews can have varied as the literature was collected and read during this time. The interviews were semi-structural and can therefore have varied in its execution in relation to questions and discussions that arose. But as the interviews were compared by pattern to each other and similar patterns was identified this influence doesn’t seem to have had any effect.

The last and probably the most important aspect is the number of participants in the study. Due to time constraints only four interviews was conducted and it could have a negative impact on the study’s validity. However, the participants had varied roles which gave a better representation of the project. And as the participants were not only from one professional group but still experienced the same problems it indicates positively on the study’s validity.

2.2.4 Conclusions and future research
Based on the interviews, a number of problems relating to the integration of UCD and agile development were identified. The main problems were planning, structure and communication. Further, a framework has been created from the literature and from which possible solutions to these problems have been identified suggested. In relation to planning, enough time before development should be given to UCD specialist in the form of a pre-sprint. Further, UCD work should later be done in a parallel track with development. A pre-sprint is necessary to for the UCD work to be timely and effective while also contributing with a clearer vision of what is to be created. The parallel tracks should be implemented so that deadlines coincide and timely support of development is possible. The parallel tracks are however contingent on UCD specialists working one (or two) sprint ahead and that a bigger design can be broken down into smaller design chunks.

The consulting firm uses their own front-end developers as a bridge between other existing developers to solve problems with the communication between developers and UCD specialists. This does not sufficiently solve the problems in relation to communication according to some participants. A possible solution according to the framework is to appoint a design lead which would be responsible for the usability requirements and making sure a focus on usability is maintained in planning meetings. Another possible solution to improve communications is to create or heighten developers’ empathy and understanding of the UCD process. This could be done by a workshop or a design studio. Furthermore, participation in the daily stand-ups by UCD specialists would create a successive delivery of design specifications by allowing developers to follow the progress and there by understand the underlying motivations more easily. Another possibility is to change what artifacts
are used as deliverables and thereby as communication means to developers. Sketches, prototypes, and user stories are all suggested in the framework as viable deliverables to communicate usability requirements, system functionality, and underlying motivations for the design.

All proposed solutions would be interesting to further examine in future research. Action research could be done to examine and evaluate the implementation of a pre-sprint and/or parallel tracks. However, this requires enough access to a new project that has not started yet or a project concerning a new version of an already existing product. It would also be interesting to examine the introduction of a design lead role or a workshop to improve the communication and understanding of the UCD process respectively. Further research could also examine the artifacts and deliverables used as communication means and their appropriateness for this purpose.
3. Method
This chapter briefly introduces case study as a research method. Further it describes the study’s design and procedure.

3.1 Design
This study took the form of a case study. Case study as a research method is most suitable when “how?” and/or “why?” research questions are being investigated. Furthermore case studies are used to investigate a current phenomenon within its real-life context and are most suitable when the boundaries between the phenomenon and the context are unclear (Yin, 2007). The case study approach was deemed fitting as this study’s research questions mainly concern how UCD specialists experience communication through boundary objects in a specific project.
The phenomenon under investigation in this case is therefore communication through boundary objects which is part of the larger phenomenon of the integration between UCD and agile methods. The context in which this phenomenon is being examined is the project where the consulting firm’s UCD specialists are involved in. Examples of possible sources within case studies are documentation, archives, interviews, observations, and physical artifacts (Yin, 2007).

3.1.1 Participants
Seven interviews were conducted with UCD specialists from the consulting firm. One participant was excluded since that participant had not been communicating through any deliverables with the developers. All the remaining participants had at one point been working in the current project and experienced communication with developers through boundary objects. However, only four out of the six were currently working in the project. Participant 1 (P1) and participant 2 (P2) was involved early on in the project and continued for about a year before leaving the project. Participant 3 (P3) had and continued to be involved in several tracks concerning different sub-services. Participant 4 (P4) had been involved in the project for little more than a year and was at the time of the interview the design lead for one of the sub-services. Participant 5 (P5) got involved in the project and in relation to a sub-service when the developers were in their second sprint. Participant 6 (P6) was at the time of the interview involved with the requirements gathering of one of the sub-services but had just prior to the interview finished working with the developers of another sub-service.

3.2 Procedure
The interviews were semi-structured and aimed at investigating UCD specialists’ experience of communication through boundary objects with developers. Furthermore they aimed to answer the research questions concerning the communication of their user understanding and their view on design rationale for this purpose. Four of the interviews were conducted face to face and two was conducted over a video call with the program Skype. The interviews were conducted in Swedish and audio was recorded for all the interviews.

A template of questions was produced with the relevant literature in mind to guide the interviews. A few examples of the main questions are presented below while the full template can be found in the Appendix.

- What documents and deliverables do you produce for the developers?
- For every document/deliverable mentioned,:
  - What is the purpose of what is being communicated?
Each interview began with acquiring consent from the participants to record the interview. They were then informed on the purpose of the study, the anonymization of collected data, and a clarification that participation was voluntary and that they could end the interview at any moment. This was done to fulfill the ethical requirements of consent, information, and confidentiality.

The interviews were later transcribed in Swedish. The analysis of the interviews began with categorizing the answers according to the research questions. Three separate content analyses were then performed in relation to each research question. Content analysis is a research tool for extracting concepts, themes, and meaning from the content of text data, such as communication material i.e. transcribed interviews (Hsieh & Shannon, 2005). Open coding was first performed on each interview to identify themes after what the passage was seen to contain. The themes from each interview were then juxtaposed to extract main themes and any eventual sub-categories. Any excerpts from the interviews presented in the next chapter have been translated from Swedish to English.
4. Results
The resulting themes from the analysis are presented below in relation to each research question.

4.1 How do UCD specialists experience the communication through boundary objects with developers?
The data collected during the interviews in relation to the research question has been categorized according to seven different themes. The themes contain answers to what and how documents and deliverables are used, what their perceived purpose is, and what their experienced advantages and drawbacks are.

4.1.1 Potential boundary objects
To uncover the participants experience with boundary objects they were asked what documents and deliverables they produced. Though with some differences due to variations of when they became involved in the design of a sub-service in the project, the answers were quite similar. The different documents and deliverables they produced, and therefore the potential boundary objects, were; (1) Personas, (2) Scenarios, (3) Effect maps, (4) Sketches, (5) Design Specifications, (6) Prototypes, (7) Evaluation summaries, and (8) User stories. In addition to their own produced material (9) demo pages, created by developers also came up as potential boundary object.

4.1.2 Different and multiple purposes
Some of the mentioned documents/deliverables have a single and clear purpose, others are used with multiple and different purposes. A few of the different types are viewed as work artifacts first and foremost and not communication material i.e. boundary objects. Personas and Scenarios were primarily viewed as working material for the initial design. The purpose of the personas was mainly seen as documentation for requirements gathering and the design process by identifying target users, prioritizing these and to convey an understanding for them. Scenarios were also primarily seen as documentation within the design phase and were used to represent possible future use situations based on personas and as a base for extracting interaction flows. Due to the focus on aiding the design process, personas and scenarios was seen as poor communication material to anyone not directly involved in the design process. The structure of the persona was created so that the creator, who also conducted the research, would later easily recall the most important elements.

This is shown by a statement from P1 “They (personas) are hard to digest, even I experience this if someone else has created it. When I’ve made a persona I know exactly what parts I’ve created it from” and “personas are hard to extract information from in the text form when they are 2-3 A4 pages long. But at the same time you make them long so that you as a designer can use them later when designing, but then they might not work as well for communication purposes”.

Prototypes were also reported to have multiple purposes. In addition to communication artifacts (which will be discussed later) it was sometimes primarily used as an evaluation tool. This reportedly affects the structure of the prototype as details in the prototype are held back so to enable more responses from the participants in evaluation. By keeping the detail of the design at a lower level, evaluation participants can concentrate on the functions being evaluated and not be distracted by the details. Further it is reported that the low fidelity gives the evaluation participants the feeling that the design is not carved in stone. It becomes apparent that as documents and deliverables like
 personas and prototypes are structured for the prioritized purpose, the suitability for other purposes diminishes

4.1.3 TAGRI principle
The TAGRI (They Ain’t Gonna Read It) principle was indirectly or directly present in all interviews in relation to personas, scenarios, and effect maps. Though all of the documents and deliverables were submitted to the project database where everyone involved including developers had access, there was some uncertainty of what developers actually saw and used. P1 and P2 expressed the experience that none of the developers read material like personas and scenarios when sent out over email. P1 and P4 further expressed that there was an overflow of material in the project database that was difficult to filter through and that this rendered the database non-functional. Although accessible to developers, P3, P4 and P6 also reports that by their experiences neither of the personas, scenarios or effect maps was used by the developers. P1 and P2 also reported that they didn’t see any use of the personas, scenarios or effect maps for the developers. Due to the experiences’ that these artifacts and deliverables are not used by developers; personas, scenarios, effect maps, and evaluation summaries are not viewed as boundary objects between UCD specialists and developers.

4.1.4 Purpose as boundary objects
After excluding personas, scenarios, effect maps and evaluation summaries from analysis as boundary objects the remaining documents and deliverables are sketches, design specifications, prototypes, and user stories. The analysis of the interviews has identified the participants’ different experienced purposes of each of these. All of the participants reported using sketches as deliverables to the developers. The sketches were reported containing requirements from the scenarios and were used to visualize these. The main purpose of the sketches was reported being to create an understanding of user flows, the interaction with the system and its functionality. It was also seen as a communication aid by grounding and structuring the verbal communication. Lastly its purpose was seen as providing instructions to the developers and to raise changes to the design.

A design specification was used by all of the participants but one. P5 reported using sketches instead since arriving at a late stage in the sub-project and that it therefore was no time to produce a design specification in its full extent. A design specification was by the other participants reported to contain sketches coupled with descriptive texts clarifying the interaction and system responses. One of the delimitations of the design specification was that it was not allowed to be interactive as it had to be printable. A design specification was also reported containing graphical design and the sketches used seemed to be hi-fi to convey the detail design.

As P3 puts it “We have created a design specification that basically describes every page of the service...with more or less finished windows in relation to design...”

The purpose of the design specification was reportedly to communicate an understanding for the interaction and the system’s response. It was used to eliminate and forgo questions but also to minimize misunderstandings and ensure the design. Further purposes were reported being to create quality and to produce criteria for verifying of the implemented code.

P1, P2, P3, and P6 all reported using prototypes as a deliverable to the developers. The prototype was mostly seen as complementary to the design specification as its purpose also was to convey an understanding for the interaction and to minimize misunderstandings. But in its advantage P2
expresses the experience of prototypes as the strongest visualization tool. P3 reported a similar experience with prototypes as conveying a better overview and a stronger feeling for use than the design specification. Furthermore P6 reported that when using hi-fi prototypes they could almost replace the design specification.

P3, P5, and P6 reported using user stories as deliverables to the developers. P3 reported using them as a complement to the design specification and prototype with the purpose of describing possible interactions to each part of the service. P5 used them as complements for the sketches that replaced the design specification with the purpose of providing the developers with finished work packages. P6 reported translating the design specification to user stories with the same purpose of providing finished work packages to the developers.

4.1.5 Neither comprehensive nor stand-alone
When the participants were asked on how they experience the communication through documents and deliverables all of them pressed that the boundary objects were not comprehensive and therefore not able to stand-alone as communication. As the boundary objects main purpose seem to be communicating an understanding of interaction, underlying and explicit motivations were excluded. Further and as P6 reports, the process of arriving at a final design is iterative and ongoing even through development. The deliverables are therefore only as comprehensive as the design is final. P3 reports that since the deliverables can contain heavy texts it will always and to some extent be ambiguous. As P4 points out, questions concerning the delivery will always arise. When trying to achieve the best possible result, the participants experienced that one would always have to verbally communicate the intentions of the document/deliverable as well. P3 therefore experienced the need to deliver in person. P6 expresses a similar need with a face to face review of the deliverables. The notion of the boundary objects being neither comprehensive nor stand-alone is further evident as the different purposes of each boundary object is closely related to being a communication aid.

4.1.6 Time consuming boundary objects
One of the emerging themes was that the boundary objects was experienced very time consuming by all of the participants. The experiences with time consumption are related to the creation of the documentation/deliverables, maintaining them, and reviewing them when delivered. The participants reportedly experience that the deliverables can become very large and heavy. This makes them difficult and time consuming to package for delivery. While wanting to explain in as much detail as possible to ensure the design, a balance has to be made between the comprehensiveness and the time spent on creating them.

P6 describes this with “it is a challenge to put together a material which is comprehensive enough but not to large”.

In relation to maintenance P4 and P6 expresses their time consuming experiences with updating sketches and design specifications. Both further expressed that if one had initial time to create a hi-fi prototype, time could later be saved on maintenance as master components can be used in prototypes. The experiences of the participants suggest that the more complex boundary objects such as design specification become, the harder they are to interpret and get acquainted to. P6 points out that as a design specification grow the ability to convey an overview of the design as a whole diminishes.
4.1.7 Direct “translation”
As the project went on, some of the participants started to seek feedback on what deliverables the developers felt they needed and how they structured their tasks from the deliverables they received. This resulted in that participants started deliver user stories as complements to existing deliverables. This was reportedly done as it would otherwise require developers having long meetings for turning deliverables like sketches, design specifications and prototypes into user stories as their usual structured tasks. In addition to lengthy meetings P5 also reports the experience with user stories having too much of a technical focus after these meetings and therefore saw the need for the UCD specialists to directly translate their work into user stories. This provided developers with finished work packages and allowed them to focus on the prioritization of the user stories in the product backlog instead of their creation. By creating user stories some of the participants could directly translate design issues, user research like interviews, and evaluations into easily interpretable deliverables for developers. It was also reported that due to time constraints in the project direct translation also occurred with sketches, as is shown by P1 statement: “You don’t put in a week’s work on creating a nice presentation of evaluation results because it is more important to get the design going and update sketches.”

4.2 How is the UCD specialists’ understanding of the user communicated to developers through boundary objects according to UCD specialists?
Three themes emerged from the interviews concerning how an understanding of the user is communicated.

4.2.1 Problems stronger than solutions
The experiences reported by some of the participants were that problems seemed to stronger communicate an understanding of the user than suggested solutions would. P1 uses the example that a video recording of the user evaluation could much clearer show the frustration of the user and would therefore convey a much stronger understanding of the user. P1 continues with “I believe that what would create more empathy is to involve the developers to a larger extent in the requirements gathering where they could meet the user...” While P2 points out that personas and scenarios contain a lot of information about the understanding of the user, evaluation results was experienced to be the strongest means to communicate the this understanding. P3 and P4 also reports that interviews and evaluation results was the primary motivation for how user stories were created.

4.2.2 “Translation”
As previously stated, the documentation and deliverables created to convey an understanding of the user is primarily viewed as working material for the design process. This coupled with the time shortage experienced by participants meant that they felt the need to directly translate the understanding of the user to actual boundary objects such as sketches and user stories. Further P1 and P2 report on the view that it is not the developer’s job to understand the user and therefore do not need to care. As described above, information concerning the understanding of the user acquired in interviews and evaluation was directly translated into user stories and sketches. Further information about the understanding of the user which could be found in scenarios and personas was also translated into sketches and user stories respectively. In P5’s sub-project user research was missing due to time limitations. This and in addition to a skewed and technical focus in user stories reportedly made P5 to get involved with the creation of user stories. This was done to restore what P5 perceived to be the user flows, which would improve the understanding of the user.
4.2.3 On demand
As it has been shown now, the documentation and deliverables directly containing information for the understanding of the user is not perceived by the participants to be received and read by the developers. Furthermore it has been reported that there are no explicated motivations for design decisions in the actual boundary objects to convey an understanding of the user and thereby the design. Instead the participants’ experiences indicate that the understanding is translated into boundary objects as the information is not seen as relevant for the developers to do their job. However, the participants report that they later verbally communicate the understanding of the user if necessary i.e. on demand by developers. This verbal communication seems to be aided by the creation of personas and scenarios as participants report that they sometimes refer back to the existence of this documentation. What emerged in the interviews was that any connections between an understanding of the user (i.e. rationale) and the actual design is internalized in the participants and only externally verbalized when requested. As P2 and P6 states, this is done to cover their backs.

4.3 What do UCD specialists think about using design rationale to communicate their understanding of the user to developers?
Four themes were identified from the interviews concerning the use of design rationale to communicate their understanding of the user through boundary objects to developers.

4.3.1 Level and formalization
Two concerns expressed by the participants were on what level the design rationale should be documented on and which type of formalization to use. In relation to what level of design rationale to use, participants expressed their concerns documenting design rationales on a detailed level. P1, P2, and P5 expressed that on a detailed level there are just too many design decisions to document them all, which would also make the documents too complex. P1 also expressed the concerns that design rationale on a detailed level would risk losing the ability to convey the overall picture. P1 also raise the issue that if verbally communicating the rationale didn’t result in that developers got a higher empathy for the user, neither would text-based rationale. One additional concern expressed by P2, was that there was no room for a design rationale in the current deliverables.

Although with some concerns about the formalization of the design rationale, half of the participants were open to the use of design rationale. P3, P4, and P6 all reported on having used a design diary to document design decisions to some extent. While P4 and P6 experienced difficulties finding an effective and suitable formalization P3 report the use of a design diary with clear criteria for when documenting decisions and changes. P3 expressed that the use of the design diary as a design rationale was both done in the argumentation and documentation perspective where it helped structure the work and function as a reference point in communication respectively.

4.3.2 Time consuming implementation
All participants voiced their concern of the time consuming work with the implementation of a design rationale. As P1 points out, small streams make great rivers. The creation and maintenance was by some participants viewed as complicated and a full time job. According to P4 a balance is needed between the time spent and the detail put into it. Some participants point out that, although optimistic about its benefits, it is not realistic as the reality of working in agile development does not always allow its implementation.
This is exemplified by one of P6 answers, “You’re always experiencing a time shortage and it therefore becomes easy to just make verbal decisions, but the bad part with that is you can’t follow your design, so it is preferable to document all decisions, but the reality doesn’t always allow that.”

P6 further points out that as time shortage is experienced, documenting the decisions become down prioritized. But P3 who was the most optimistic towards using a design rationale to communicate the understanding of the user expressed the experience of it only being time consuming in the beginning.

4.3.3 Relevance
In relation to concerns about what level a design rationale would be documented on, concerns about knowing which rationale that would be relevant to developers in the future was also expressed. While participants voiced concerns about the time consuming work with documenting every design decision and the need for a balance. They also expressed concerns with filtering relevant design decisions from irrelevant ones from the developers’ perspective. As mentioned, P3 has used a design diary as a design rationale in both the argumentation and documentation perspective. While deciding which design decision to document was based on the argumentation for aiding the creation process, the filtering seems to be a reactive process based on requests from developers. Furthermore, P4 report that some decision can seem obvious and therefore a waste of time to document. P6 points out a similar opinion with “If I don’t know if it’s going to be some use for others than myself, the documentation will suffer.” P2 speculated that developers might not be interested in the rationale behind details in the design but might show interest with high level rationale. Furthermore, P2 points out to the TAGRI-principle with “people hardly read anything and they want as little text as possible...so you would have to make it very short and let them come back to you for the entirety of the reasoning.”

4.3.4 Advantages
Although some participants seemed negative about the implementation of a design rationale, all but one still acknowledged some benefits of its use. These can be divided into benefits regarding the argumentation or documentation perspective. In the argumentation perspective P3 and P4 reported that it can be useful for the creation process by structuring their work. In the documentation perspective P1, P2, P3, P4, and P6 saw the benefits of design rationale as a communication aid. P6 point out that there can be a sense of security with the use of design rationale and further states, “It’s very important for us that we base our decisions on empirical evidence...and it has to be good for the communication to be clear with that. “ P3 also report that benefits of a design dairy is that it can function as a memory aid and enables backtracking. P3 also experienced it could help in maintaining the overall picture of the design.
5. Discussion

The discussion surrounding the results is structured according to the research questions although there is some overlap between the answers. The main discussion points relating to the first research question are the heavy weight on functional repositories (e.g. project database, product backlog, personas) created in distributed teams, the potential use of hi-fi prototypes to solve time-consuming maintenance of deliverables, and the benefits of direct translation. In relation to the second research question the discussion focuses on the importance of active involvement of all members in key phases, the need for a balanced integration, and the potential of using design rationale to aid the access of information in the personas. The discussion relating to the third research question will mainly treat how the participant’s concerns relate to the generic structure of design rationale and how each concern could possibly be solved. A discussion about the method will conclude the chapter.

5.1 How do UCD specialists experience the communication through boundary objects with developers?

As stated in the background agile principles value face-to-face interactions and very little documentation. But when the agile team becomes distributed, communication and the grounding of information become more difficult. This puts a heavier load on boundary objects as references in other communication. From the interviews it was concluded that all documentation and deliverables were distributed through a project database or a product backlog. The database could in a sense also be seen as a boundary object but was excluded from analysis since the database itself did not contain any information except other boundary objects. The scrum team’s backlog was also excluded from analysis since it only contains user stories. What needs to be noted is that developers could by prioritizing the user stories in the backlog inform UCD specialists about which part of the design the UCD specialist should focus on in the next iteration. However, since the study is delimited to UCD specialists’ communication to developers and not vice versa the backlog was still excluded from the results. In addition, only one participant reported that the backlog was being used this way which also spoke for the exclusion of the backlog. The project database and the backlog can through a boundary object perspective be seen as the boundary object type repository, identified by Star & Griesemer (1989). What becomes clear is that using distributed agile teams puts a heavy weight on the functionality of repositories and the need for easy access in them. The difference between the project database and the backlog seems to be that, while the functionality and easy access of the backlog is apparent as it is deeply integrated in the agile method, the project database failed leaving some documents and deliverables unseen.

Personas and scenarios were reported having multiple purposes, being both a tool in the design process and communication material. The main purpose for this documentation was however being a tool for the design, where the main benefits seemed to come from the process of creating them. As they were viewed as a UCD specialist’s tool, they were often created by a specific UCD specialist for that specific UCD specialist’s use. This meant that the structure of the information in the persona was formed in a way just to aid the UCD specialist’s work. As the purpose, that of being a design tool, shapes its structure it seems a negative effect renders them ill-suitable for communication to developers. Although the analysis excluded personas and scenarios as boundary objects in this study they could easily be interpreted as ones in other contexts. From a strictly conceptual boundary object perspective this documentation could be seen as the repository type of boundary objects. As we
previously saw with the project database, such boundary objects needs to enable easy access of information in order to be effective. For the purpose of being a design tool, the enabling of access to the information becomes a non-issue in this study as the designers creating the documentation was the same one using it. It seems that one way to look at it is that the designers internalize the index key for accessing and retrieving the information.

One solution to this problem can be found in the framework presented earlier, specifically in the category of collaboration and communication. According to Chamberlain et al. (2006) all team members need to participate in key phases so not to create communication problems later on. One of the projects under study in their article reportedly involved the whole team in creating user journey (similar to personas and scenarios) as not doing so in the past had created problems. This option is however dependent on the agile team not being distributed. This solution does not however create a more communicable structure in the general sense. If working in a distributed agile team, more focus has to be spent on making the personas and scenarios easier to interpret and access information in. As mentioned earlier in the background, Adlin & Pruitt (2006) describe the personas lifecycle and points out that one of the most difficult tasks is the transition from creating a persona to using one in an organization. They stress the fact that personas have to be presented progressively in small and easily interpretable chunks. More importantly, the personas have to be structured after the intended receiver. This means that when creating a persona it must be clear how to access and retrieve information, not only by the main intended user but all other intended and possible users as well, including developers.

The experience of time consuming boundary objects related both to the creation and maintenance of them. In particular the design specification was reported time consuming in its creation because of heavy texts and in maintenance due to the need to update the sketches individually. P6 mentioned that sometimes, and if there was enough time, a hi-fi prototype could be created and function as a replacement for the design specification. What is represented in text in the design specification could possibly also be simpler represented in a hi-fi prototype. While the design specification with its static sketches allows the representation of specific flows within the design, coupling the prototype with user stories could just as easily convey this information. As both the design specification and prototype was seen as neither comprehensive nor stand-alone, verbal communication was anyway needed to clarify the artifacts. The possible benefits of using a prototype for this purpose are that the prototype is reportedly easier to interpret and gives a clearer overview of the design. Furthermore, it was seen as a stronger communication material because of its interactive qualities. Even though the hi-fi prototype was still considered time consuming in its creation it was considered less time consuming in its maintenance. This was due to the possibility of updating master elements which would automatically update every instance of that element throughout the prototype. The framework presented in the background shows that lo-fi prototypes are the most common technique used by UCD specialists in agile development (Hussain et al., 2009b). Both from the framework and as pointed out by P3 lo-fi prototypes are mostly used for early evaluation of the systems and its functions. They take the form of lo-fi instead of hi-fi to elicit more responses from the users in the evaluation and to save time in its initial creation. The hi-fi prototype should therefore be seen primarily as a communication artifact and deliverable but could of course be used as an evaluation tool for the phase between early evaluations and beta testing (e.g. heuristic evaluations).
As two participants became co-located with the rest of the development team they started asking for feedback on what deliverables they should produce and how the developers structured their work. This resulted in user stories being added as deliverables. This shows the importance of the empirical process control which lies as a foundation for Scrum. The three cornerstones of the empirical process control are visibility, inspection, and adaption. Although not contingent on being co-located, the inspection of what developers needed to structure their work lead to an adaption of the deliverables to include user stories. As Convertino et al. (2009) states, the understanding of what is critical to share increases as the team becomes co-located. From a boundary object perspective user stories can be seen as what Star & Griesemer (1989) calls standardized forms as they are structured after user role and goal (e.g. “as a [user role] I want to be able to [goal]”). User stories thereby show an advantage of using the boundary object type standardized forms as communication means. User stories, created by UCD specialists, appear to provide the developers with a more direct translation of the UCD work than other deliverables like design specifications and prototypes as they already are an integrated part of the agile development. According to Bayer et al. (2004) user stories can be seen as relatively detailed definitions of systems functions which are derived from user goals. In addition Brown et al. (2008) report on the advantages with user stories as they can connect the design to user categories and user goals. Despite these previously reported benefits none of the participants expressed the view of user stories ability to communicate an understanding of the user. Translation also occurred from scenarios and evaluation results to sketches. As deliverables was seen to be neither comprehensive nor stand-alone sketches could be presumed to be a preferable communication aid. This is in line with Brown et al. (2008) who have identified sketches to be well adapted for focusing attention, grounding communication, recording design decisions and to explore relationships within design.

5.2 How is the UCD specialists’ understanding of the user communicated to developers through boundary objects according to UCD specialists?

One of the main themes in relation to how an understanding for the user is communicated was that problems were experienced to communicate a stronger understanding of the user. P1 expressed the need for developers to be more involved in the requirements gathering to get a better understanding of the user. This is also found in the literature as Chamberlain et al. (2006) press the point that all team members should be involved in all the key phases of the project so not to create communication problems later on. This is further illustrated with Patton (2002) where he reports the use of a workshop with developers and stakeholders to performed user research and requirements analysis. Patton claims this enabled a distribution of knowledge for the user throughout the team and that the resulted prototype could function as a communication aid later on. The involvement of all team members in key phases can also be connected to the agile manifesto and the accompanied twelve principles. As principle number 4 states, business people and developers must work together daily throughout the project. As UCD specialists are highly active in the requirements gathering and analyzing in the current project they could in this case and in relation to the agile principle be interpreted as business people. The key words of the principle is “throughout the project”, which means that they should to some extent be involved and participate in requirements analysis. Although it was evident in the pre-study that a pre-sprint (sprint 0) dedicated to UCD work is needed, it seems that this pre-sprint should not exclude developers completely.
Time constraints seemed to lead to the direct transfer of evaluation results and personas to sketches and user stories. While this translation has been discussed in relation to the previous research question, it will be here discussed in a different meaning since it can be seen as a consequence of a deeper issue. Blomkvist (2005) argue that an integration of the two methodologies can be seen from both an abstract and concrete perspective. The concrete perspective in this case would be that information and artifacts are skipped due to time constraints, but the reason for this time constraint appears to be due to the lack of a proper integration in an abstract sense. As Blomkvist points out, for an integration to be successful, UCD does not only need to be integrated in the agile methods, agile methods should also be integrated in UCD. This should be done to allow a better integration of the underlying values. Not allowing enough time to adequately present the results of an evaluation could be seen as a failure to integrate the underlying values of UCD in agile development. The lack of a deeper integration on the abstract level also becomes apparent as P1 and P2 claim that it is not the developers’ job or responsibility to care about and try to understand its user. According to the agile principles the highest priority is to satisfy the customer. This should obviously include developers knowing for who they are actually building a system for.

The analysis of the interviews showed that the UCD specialists’ understanding of the user is not considered by themselves to be communicated through boundary objects. Although an understanding of the user has been shown to be communicated through user stories (discussed previously), the participants experienced that their understanding of the user was verbally communicated and only on demand. This is in line with the sixth agile principle which states that the most efficient and effective way of conveying information to developers is through verbal communication. When verbally communicating their understanding of the users, the participants reported using boundary objects as communication aids but also referring back to the existence of personas and scenarios. The use of boundary objects such as user stories and sketches as communication aid is, as previously mentioned, supported by Brown et al. (2008). While sketches is suitable for visualizing and grounding the communication, user stories are well suited for making problems apparent and connecting designs to user groups. As Modi et al. (2013) states, the importance of boundary objects as communication aid grows as teams become distributed.

While personas and scenarios might be mentioned in verbal communication, it appears that the participants only uses the existence of these to keep their backs clear if a design decision is questioned. There seem to be no physical use of personas and scenarios for aiding the communication with the developers. In relation to the view discussed earlier where personas could, in a different context be viewed as repositories (type of boundary object), the UCD specialists’ function seems to be that of an index key for the information it contains. This is due to the fact that any connections between the design and personas mainly appear to be internalized within the UCD specialist. On a higher level it is shown that user stories have the potential of connecting the design and the user. However, as the participants’ understanding of the user surely is more gradated than a simple user story the persona should still have an important role to play.

A design rationale as an additional boundary object could possibly be used for the purpose of bridging the design and the personas by externalizing the participants’ internal connections. This would, hypothetically, not only relieve the UCD specialists of holding in the connections but also solve the access related difficulties with the personas. The solution of a design rationale is not necessarily one created for the developers’ ease of access (i.e. retrieval). A design rationale in the
argumentative perspective could also be used to aid UCD specialists in the design process and therefore possibly help clarify what is essential to motivate. As Adlin & Pruitt (2006) suggests, the persona should be adapted for the intended receivers. The persona could therefore and later be re-structured from the design rationale as to possibly provide a better aid for communication with developers.

5.3 What do UCD specialists think about using design rationale to communicate their understanding of the user to developers?

It should first be mentioned that the concept of design rationale in all its meaning was not widely known to the participants. Before the interviews started the participants was asked of their knowledge of design rationale. Since it was not widely known, all of the participants were given a small verbal summary of the three perspectives (Argumentation, Documentation, and Communication) of design rationale presented by Shipman & McCall (1997). Some of the participants however used or had used a design diary, which according to Moran & Carroll (1996) can be seen as a design rationale in the documentation perspective. The results of the interviews show the concerns and possible benefits experienced by the participants and may not give an exact account of what is actually possible with design rationale in the general sense.

As Lee (1997) shows, the generic structure of design rationale is composed of three layers. These layers are the design intent layer, the decision layer, and the design artifact layer. Participants showed being positive to the use of the design intent layer, exemplified by the reported use of effect maps and the design artifact layer by using sketches, prototypes, and user stories to show elements of the design and how they related to each other and the user. However, the reported reluctance for using a design rationale can be connected to Lee’s decision layer. Concerns relating to the decision layer were that there seemed to be too many design decisions to document them all and that they needed to be filtered. This was further expressed with concerns about knowing what is relevant and the time consuming effort of documenting all design decisions.

The view of creating a design rationale as too time consuming is most likely caused by the fact that UCD specialists are not used of working in a high tempo agile project and that agile methods do not always support the creation of large documentation. However, Sauer (2003) argues that agile methods do not necessary make documentation such as design rationale impossible or obsolete. Sauer claims that an event-based and semi-automatic design rationale is suitable and can be used in agile projects. Although Sauer’s example of the use of the event-based systems handles management decisions he claims that it could easily be turned into an argumentative system. Although, as the system is only semi-automatic, it could be too time consuming in its initial use when argument prototypes need to be created for each specific domain.

The second concern relating to the decision layer was that of knowing which decisions are relevant for the receiver, which in this case would be the developers. The relevance of decisions for the receiver seems like a common problem and Falesi et al. (2013) calls this problem information predictability. The concerns about the decision layer become more apparent as P1 and P2 expressed the view of the rationale being irrelevant for the developers’ work and that it’s not their job to care. The more positive responses to the use of a design rationale in general came from the participants who were still active in the project when UCD specialists in sub-projects became co-located with developers. It would seem that as participants become co-located with the developers their
awareness of the developers’ activities grows and therefore can more easily predict the relevance of information. This is in line with Convertino et al. (2009) as they claim that the understanding of what is critical to share increases as the team becomes co-located.

For the implementation of design rationale to be successful it has to be well integrated with both the design process and the tools used by UCD specialists (Moran & Carroll, 1989). Furthermore, it has to be well integrated in the development process so to enable the efficient retrieval for developers. As previously mentioned the design rationale in the argumentative perspective could possibly aid the information predictability as it helps UCD specialists structure the design process. As for the use of a system for the developers’ retrieval, the question still remains if design rationale would be considered a stand-alone and comprehensive documentation or still needing verbal clarification. If still needing verbal clarification the structuring of decisions for retrieval would only be additional work.

5.4 Method discussion
The study was delimited to examine the UCD specialists’ experiences with communication through boundary objects to developers within the specific project. This delimitation has two weaknesses. First of all it only focuses on boundary objects directed at developers. Since the project and its constituent sub-projects was further divided into requirements and development tracks, some communication through documents and deliverables was excluded since it was not directed to and received by developers. The second weakness is that the developers’ experiences were excluded. This becomes a rather large issue as they are the receivers of documents and deliverables and therefore “decides” how well the intended information actually is communicated. It would also have been very interesting to validate and/or compare the different experiences of the different roles. It would have been interesting to see if some of the deliverables were more appreciated by front-end or back-end developers. The study was initially planned to include the developers’ experiences with the communication through documentation and deliverables created by the UCD specialists. Since access to the developers in the project was more restricted, a questionnaire was created aiming at examining how developers experienced communication through the identified boundary objects. The questionnaire even got sent out through a mediating party in the project. But as the frequency of answers to the questionnaire was too low it was decided to be excluded.

The study was also delimited to only examine boundary objects created by UCD specialists. This resulted in that the interviews only concerned these boundary objects. Although one boundary object created by developers came up in one or two of the interviews, surely more instances of this would have occurred if the interviews had been differently constructed so to examine all possible boundary objects from both sides. This gives a rather slanted perspective since any further and eventual boundary objects created by developers are missing from analysis.

Since the project and its constituent sub-projects had such a clear distinction between requirements gathering tracks and development tracks, some of the involved UCD specialists in the projects had to be excluded from the study as they had not been communicating with developers through boundary objects. This meant that only six UCD specialists participated in the study. This rather low number may have affected the validity of the study. But this is to some extent compensated as the UCD specialists participating came from several sub-projects and still expressed somewhat similar experiences.
The data sources in the study were limited to interviews and the validity of the study could therefore be suffering. As previously mentioned triangulation is an important concept within case studies. It would therefore been preferable to observe the use of boundary objects in the project. But due to limited time and access this was not possible. It would also have been preferable to perform analysis of the documentation and deliverables themselves from a boundary object perspective. But as the project was restricted to share any documentation with outsiders this was not possible.
6. Conclusions and future research

In relation to research question one “How do UCD specialists experience the communication through boundary objects with developers?” it has been shown that not all documents and deliverables created by UCD specialists can be considered boundary objects in relation to developers. It first of all shows the importance of well functioning repositories with easy access. If not so, they risk losing their boundary qualities. It is shown that for personas to eventually be considered boundary objects they need to be structured to support the access of information for developers. One conclusion that can be drawn from the study is that the purpose of the actual boundary objects is to communicate an understanding of the interaction with the system and not necessarily the understanding of the user. The boundary objects created for this purpose is considered neither to be comprehensive nor stand-alone and therefore holds the additional purpose to be used as communication aids. As the participants describe the boundary objects to already be time consuming enough, trying to make boundary objects to be comprehensive is an almost impossible mission in agile development. Although time consuming in its initial creation the use of a hi-fi prototype could save time on maintenance and updating as the design process continues throughout the development. One further conclusion that can be drawn is that UCD specialists directly translated their work into sketches and user stories. On the one hand this shows the importance of inspection and adaption of the process as UCD specialists started to create user stories as design deliverables. One of the reasons for this translation and the one to sketches were to save time. Another was that these seemed to be more successfully integrated in the developers’ work process. While sketches can be considered to be strong communication aids, user stories show an advantage of helping developers structure their work and connect the user to the design and its elements.

In relation to the second research question “How is the UCD specialists’ understanding of the user communicated to developers through boundary objects according to UCD specialists?” three themes were identified. Although problems were seen to stronger communicate an understanding of the user, any documents and deliverables containing these were not experienced to be received by developers. This leads to the conclusion that for developers to truly understand the user, they would have to participate to some degree in the user research. The alternative, as participants reported, was to verbally communicate their understanding of the user to developers on demand. This makes the connections between the design and the user internalized within each individual participant. However, higher-level connections become externalized by coupling user stories with sketches and/or prototypes.

In relation to the third research question “What do UCD specialists think of using design rationale to communicate their understanding of the user to developers?” it can be concluded that participants were positive of the design intent layer and design artifact layer of design rationale. However, concerns were expressed about the decision layer and the time consuming activity of documenting this. One solution might be the event-based and automatic design rationale proposed by Sauer (2003). In relation to the decision layer participants also expressed the concern about not knowing which decisions that would be relevant to developers. By using an argumentative design rationale to aid the design process it could be possible to also aid the filtering of relevant design decision as they are made explicit.
6.1 Future research

Future research in relation to the communication through boundary objects should probably investigate how these are interpreted and used by actual developers. The comparison of experiences between UCD specialists and developers could prove to be a fruitful venture to better the integration between UCD and agile development and add valuable insights to the existing research. Future research could also examine more specifically how to re-structure the documents used for conveying an understanding of the user to better suit the agile development and its members.

Although the investigation of communication through boundary objects should have a central role in distributed agile teams it is not necessary the most important one for an integration of the methodologies in the larger perspective. The initial aim of the study was to implement changes to the planning of the project to better facilitate the information and role of the UCD specialist. Another and later goal was to bridge the gap between the UCD specialists and developers and their understanding of the processes behind each other’s work through a workshop. Due to complications these goals proved to be too time consuming to implement in the studied project. This in turn shows another interesting aspect to study. Future research could therefore direct its attention to what obstacles and limitations arise from a higher organizational perspective.
7. References


Agile Manifesto (2001) Agilemanifesto.org (hämtad 2013-12-30)


8. Appendix

Interview template

What document and deliverables do you produce?
Which ones do you deliver to the developers? (front-end, back-end?)
Do you know if the developers actually read and use them?

For each document/deliverable:

What is the purpose with the communication through the document/deliverable?
What is communicated?
Is the purpose of the content the same as the communication?
How do you experience the actual communication through the document/deliverable?
(How do you perceive developers interpreting the communication?)
(How do you think the developers using the documentation?)

What benefits and drawbacks do you experience with the communication through documents/deliverables?
Do you use the documentation/deliverable as references in other communication?
If yes, when and how?
Do you experience it aiding other communication?

What is your overall view on communication through documentation/deliverables in the project?

How do you experience it?
Is it working?

How do you take design decisions? How do you base them?
Is it documented and if yes, how?
How do you convey your understanding of the user in the documentation/deliverables?
Are design decisions motivated in the documentation?
if yes, how?
Is any specific language or system used?
If used, for what purpose and in what perspective?
Argumentative?
Documentation?
Communication?

What is your view on design rationale?
theoretical?
Practical?
Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet – or its possible replacement – from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/

© Johan Persson