UNIVERSITY OF OSLO
Department of Informatics

Agile/UX Integration:

how user experience-related practices and processes are integrated with Agile development processes in real-world projects

Master thesis

Anna Dahl

February 1, 2012

ABSTRACT

The research presented in this thesis provides empirical data on how work practices and processes related to user experience (UX) are integrated with Agile software development processes in real-world projects. Agile processes do not inherently provide rules or guidelines for how or when UX-related activities should be conducted, and the Agile/UX integration field of study investigates how the two may best be combined.

Five case studies have been conducted as part of the research. Two of the cases focus on integration in Scrum projects, two focus on integration in Kanban projects, and one focuses on integration in a "general Agile" project. All of the cases involve multi-year projects with mature Agile teams. As the Kanban process is fairly new in a software development context, little empirical research exists in this area in general, and previous empirical studies of Agile/UX integration in Kanban projects have not been found. Detailed descriptions of all the cases are presented in the Results chapter.

A systematic search of relevant literature shows that suggested approaches to Agile/UX integration mostly follow the "parallel track" model, in which developers and UX designers work in separate, parallel tracks. Designs and specifications are created one or more cycles/sprints ahead of development, and completed features are validated and tested one or more cycles/sprints after development. A short up-front analysis/design phase is usually recommended. Effective use of the model requires using "lightweight" UX techniques like paper/low-fidelity prototype testing, RITE testing, lightweight Personas and informal cognitive walkthroughs.

The study results suggest that the parallel track model is better suited to describe integration in settings where fixed time boxes are used, as in Scrum, than in settings where time boxes are flexible, as in Kanban. Future work in this area may benefit from developing and using additional, alternative models to describe and study integration.

A set of general advice targeted at process designers based on study results is presented in the last chapter. An important element to be considered when designing an Agile process is to allow for design iteration based on feedback from UX testing and evaluation. Ideally, the process should have a built-in loop at regular intervals for gathering feedback and for acting on the results.

ACKNOWLEDGEMENTS

Many have contributed to this thesis in various ways. First, I would like to thank my supervisor Amela Karahasanovic for valuable feedback along the way. I am also very grateful to all of the participants in the study, who kindly took the time to be interviewed, and to read and provide feedback on the draft case reports.

I would also like to thank close colleagues and friends for inspiring discussions and input throughout the process. Finally, my special thanks to my parents and to my boyfriend, for their constant support and encouragement.

Oslo, January 2012

Anna Dahl

Table of contents

List of Tables

List of Figures

1 INTRODUCTION

Recent years have seen a surge of popularity for Agile software development processes. Where these were once considered radical and controversial, they are now mainstream, and in some areas – such as the development of web-related systems – they appear to be the de facto standard. The Agile process Scrum is now used for large, multi-year projects with up to hundreds of developers, as well as in the small, single-team projects from which it originated.

Using dedicated people with specialized skills to handle issues related to user experience is also becoming increasingly more common. For commercial products, the user experience is often what sets the product apart – and ahead – of its competitors. For systems developed for in-house use, it is usually important to achieve increased efficiency in internal work processes, and part of the effort to achieve this includes optimizing the system to fit the needs of the users.

As we build ever more complex systems with increasingly sophisticated user interfaces, we need to make sure that the processes we use support the task at hand. Agile software processes do not inherently provide guidance or rules for how roles, work processes and practices related to user experience should be integrated or conducted. Many traditional work practices from the fields of human-computer interaction, user-centered design and similar are not, at the outset, suitable for use in an Agile environment. Thus, adaptations need to be made on both sides, and it may be challenging to figure out how this may be done in practice.

The topic of Agile/UX integration has been discussed in the professional community for several years, but advice based on empirical research is scarce. Although there is a growing body of literature in this area, the amount of research has so far been limited. A recently published systematic review on the topic concluded that there is a clear need for more empirical studies (Silva et al., 2011 p. 83).

1.1 Objectives and research questions

The study conducted as part of this thesis is an attempt to broaden the empirical base for advice to practitioners, as well as for future studies within the field of Agile/UX integration. Two research questions are sought answered:

1) *How can UX-related work practices and processes be integrated with Agile software processes?*

2) *How are UX processes and UX-related work practices integrated with the Agile processes Scrum and Kanban in practice [in a web-related environment]?*

1.2 Contributions

This research has

1. broadened the empirical base for research in the Agile/UX integration field of study
2. provided empirical data on Agile/UX integration in Kanban projects
3. suggested additional perspectives to be considered within the field
4. provided general advice for practitioners (process designers)

1.2.1 Empirical data on Agile/UX integration

In their recent systematic literature review, Silva et al. (2011) find that there is "a clear need for more empirical and/or experimental studies" in this field (ibid., p. 83). This research has provided rich and detailed descriptions of Agile/UX integration practices in five projects, broadening the empirical base for further research in this area.

1.2.2 Empirical data on Agile/UX integration in Kanban projects

The results from two of the cases in the study may be of particular interest to practitioners and researchers alike, as these describe integration in practice in projects using Kanban. To the best of the author's knowledge, no other empirical studies on this have been conducted so far, and the topic has not been discussed in any of the papers found in the systematic search (described in chapter 3, Related Work). This is not surprising, as the Kanban software process is one of the newest members of the Agile family – the first book was published in 2010 (Anderson, 2010). However, based on the amounts of interest this process is currently receiving at conferences and in online discussion forums targeted at practitioners, it appears that the adoption rate is fairly high and increasing. In the context of Agile/UX integration it is also especially interesting to note that in both cases, the resident UX specialists have been heavily involved in designing the process.

1.2.3 *ADDITIONAL PERSPECTIVES*

Most of the empirical studies conducted so far have also mainly investigated the integration from a UX perspective, that is, how UX work practices and processes are adapted in order to fit the development process. This study attempts to add the opposite perspective: how the development process is adapted in order to support UX-related work. The case study descriptions thus detail both kinds of processes, and in all cases it was found that adaptation takes place both ways.

Also, the research has shown that the well-known “parallel track” model (section 3.2.1.5/Appendix C) may not be entirely suitable for studying and designing integrated processes in settings where fixed time boxes are less emphasized, as in the Kanban cases. It is suggested that an alternative perspective may be fruitful, and a starting point for developing such a perspective is presented (section 7.2.1/Appendix D).

1.2.4 *ADVICE FOR PRACTITIONERS*

Ultimately, all research in this area is performed in order to gain sufficient knowledge to be able to provide valuable advice to practitioners. It is clear that this field of study is still very young, and no coherent framework or generally accepted process model has been developed so far. The existing advice for practitioners found in the related literature is mostly targeted at UX specialists. It is the view of this author that there is also a need for advice targeted at process designers, regardless of role. This process-oriented view is reflected in the advice provided in chapter 8.

1.3 Chapter Overview

Chapter 2 Background	This chapter presents background information relevant to Agile/UX integration and the present study. This includes discussions of Agile as a concept and the roots of Agile, short descriptions of the Scrum and Kanban processes, a short overview of the "UX" concept, and an overview of some general issues that affect integration.
Chapter 3 Related Work	This chapter presents the results from a systematic search for related literature. Although there is still little academic research on this topic compared to other Agile-related issues (technical, organizational, managerial), a body of work now exists that explores the topic both in a theoretical and an empirical perspective. Relevant empirical studies are presented in section 3.2.2.
Chapter 4 Method	This chapter presents the method used in the empirical study. The background and rationale for using the case study approach is presented, and a detailed description of the research design and how the overall study was conducted is included. Researchers planning to conduct a study within this field may be especially interested in section 4.1.2, in which examples of various methods used in previous studies are given.
Chapter 5 Results	This chapter presents the results from the five empirical case studies. Section 5.2 (case A) and 5.3 (case B) describe Scrum projects, section 5.4 (case C) and 5.5 (case D) describe Kanban projects, and section 5.6 (case E) describes a "general Agile" project. A cross-case summary may be found in section 5.7.
Chapter 6 Validity	This chapter presents issues that might affect the validity of the study. In section 6.1, an overview of criteria for judging interpretivist research is presented. In section 6.2, specific issues related to the study are discussed.
Chapter 7 Discussion	This chapter discusses the results from chapter 5 in light of the propositions presented in section 4.2.2. In section 7.2, some thoughts on the emergent theme of teams switching from Scrum to Kanban are discussed.
Chapter 8 Conclusions and further work	This chapter summarizes the results and suggests further research. General advice targeted at process designers is presented in section 8.3.
Appendix A	This appendix presents the Agile manifesto.

Appendix B	This appendix provides a description of Cockburn's three levels of software method understanding, as revised by Boehm and Turner (2003a).
Appendix C	This appendix presents a version of the Parallel track model based on the Sy (2007) version and the Silva et al. (2011) version.
Appendix D	This appendix presents a descriptive model of the life cycle of a user story, based on the findings in the empirical study.

2 BACKGROUND

In this chapter, background information relevant to the topic of Agile/UX integration is presented. In section 2.1.1, Agile as a concept and its roots are presented. In section 2.1.2, the Agile process Scrum is presented. In section 2.1.3, the Agile process Kanban is presented. In section 2.1.4, the concept of user experience (UX) and related terminology is discussed. In section 2.1.5, general issues related to Agile/UX integration are presented.

2.1 AGILE

The popularity of Agile development processes surged in the middle of the 00's, and it suddenly seemed that Agile was in focus everywhere – blogs, magazines, journals, conferences, books. According to Dybå and Dingsøyr (2008), a 2005 survey in the USA and Europe showed that 14% of companies were using agile methods, and that 49% were aware of them and interested in adopting such methods. A large 2009 US survey showed a 35% adoption rate (West and Grant, 2010)[1]. The same year, ScrumAlliance.org reported that there were now over 50,000 certified Scrum Masters[2] in the world. In a 2010 research report reporting on the results of a large survey of more than 30 000 IT personnel from around the world, it is stated that "Agile delivery methods continue gaining popularity. Over 60% of surveyed companies leverage agile, although most are still experimenting, utilizing agile only in a portion of new IT projects. This approach allows organizations to adopt agile gradually." (Aksu and Li, 2010 p. 4).

In order to investigate and understand how UX-related work practices and processes can be integrated with Agile software processes, one needs to be familiar with the background and goals of both. In some cases the interests of both worlds coincide, but in others the difference in perspective and beliefs create problems for those trying to integrate them. In this section, we will therefore take a brief look at the common traits of Agile software processes and the philosophies underlying them.

[1]Forrester Research report, available from http://www.forrester.com/rb/Research/agile_development_mainstream_adoption_has_changed_agility/q/id/56100/t/2 [last accessed 20th Oct 2011]

[2]According to the certified Scrum trainer Geir Amsjø's blog "Det smidige hjørne" 21st of April 2009 - http://scrummaster.no/?p=240 (Norwegian) [last accessed 28th Nov 2011]. The data source could not be located.

2.1.1 *AGILE PROCESSES: COMMON TRAITS*

There are many flavors of Agile software development processes. Abrahamsson et al. (2002) describe and discuss Adaptive Software Development (ASD), Agile Modeling (AM), the Crystal family (several processes), Dynamic Systems Development Method (DSDM), eXtreme Programming (XP), Feature-Driven Development (FDD), a stripped-down version of the Rational Unified Process (RUP) called dX, and Scrum. In Boehm and Turner (2003a) the Lean Development (LD) flavor has been included, and recently the first book about software development using Kanban was published (Anderson, 2010).

Abrahamsson et al. (2002) provide a very useful, straightforward definition of what makes a development method an Agile one:

"[...] this is the case when software development is

- incremental (small software releases, with rapid cycles)
- cooperative (customer and developers working constantly together with close communication)
- straightforward (the method itself is easy to learn and to modify, well documented), and adaptive (able to make last moment changes)" (ibid., p. 98)

Boehm and Turner (2003a) discuss each method in terms of their levels of concern (organizational scope for which the method provides specific guidance), life cycle activities and sources of constraint. They state that the fewer constraints a method puts on the implementer, the more agile it can be considered. The discussed methods are compared according to their number and strength of constraints, and given an "agility rank" based on the comparison. This perspective is also found in Kniberg and Skarin (2010), who compare a set of methods (using the term "process tools") according to the number of rules of each, and place them on a scale from "more prescriptive" to "more adaptive". In Figure 1 below, these two sources have been used to place the some of the most well-known methods on a comparative scale of Agility, with a special emphasis on Scrum and Kanban:

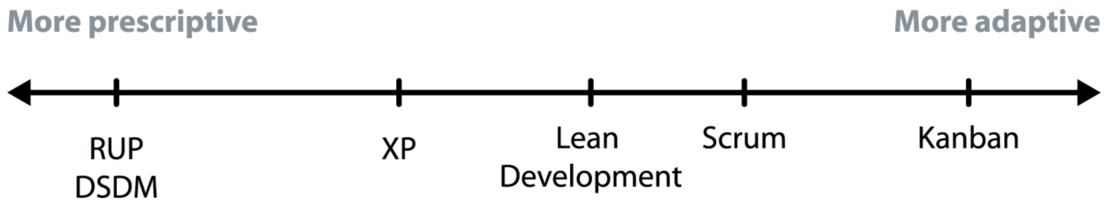


Figure 1: Agile processes compared on the prescriptive vs adaptive scale, based on Kniberg and Skarin (2010) and Boehm and Turner (2003a)

As mentioned in both, less prescriptive does *not* mean less difficult. The current version of the official Scrum guide recognizes this: "Scrum is [...] simple to understand [...] extremely difficult to master" (Schwaber and Sutherland, 2011). Fewer prescriptions means added flexibility to take contextual variables into consideration (project size, type, competence etc.), but at the same time process designers need to be able to make the *right* adaptations for the right reasons – with less guidance.

2.1.1.1 Roots of Agile

The roots of Agile software processes can be roughly divided in two groups: Iterative and Incremental Development (IID), and Lean manufacturing. The first can be seen as an attempt to handle the complexity of software development, while the latter can – somewhat simplified – be said to be a way of handling market complexity (delivering what the customer wants, when the customer wants it).

As described by Larman and Basili (2003), IID has been applied in software projects as far back as the mid-1950s. As the term implies, the primary characteristic of this approach is that the product is developed *iteratively* – multiple development iterations that serve to improve product quality – and *incrementally*, or "piece by piece". Larman and Basili describe several IID projects from each decade, and although the actual methods used vary, "all the approaches had a common theme – to avoid a single-pass sequential, document-driven, gated-step approach" (ibid., p.47).

The approach that IID seeks to avoid, is the one that is often labeled "traditional", alternatively "plan-driven" and/or "document-driven". This represents a different strategy for handling the complexity of software development, but one that is based on engineering disciplines and large aerospace development (Boehm and Turner, 2003a) rather than the smaller, software-intensive projects that Agile grew from (Schwaber and Beedle, 2002, Patton, 2008b). The basic idea is that the product as a whole is developed in sequential steps, from requirements to finished code. For each step, well-defined work products (like requirements-, design- and specification documents) are produced. Before work is started on the next step, the results from the previous step are validated and verified.

This approach makes sense in a number of settings, especially when modified to include elements of incremental and evolutionary processes. Boehm and Turner (2003a, 2003b) have identified five critical factors involved in determining the relative suitability of Agile or plan-driven methods in a particular project setting[3]:

- ***Size***. Plan-driven methods evolved to handle large products and teams, and are hard to tailor down to small projects. Agile methods are well-matched to small products and teams, and are difficult to scale up on account of the reliance on tacit (non-written) knowledge.
- ***Criticality*** (potential loss due to impact of defects). Plan-driven methods evolved to handle highly critical products and are hard to tailor down efficiently to low-criticality products. Agile methods represent potential difficulties for safety-critical products on account of simple design and lack of documentation.
- ***Dynamism***. Agile hallmarks such as simple design and continuous refactoring are excellent for highly dynamic environments (frequent change), but a source of potentially expensive rework for highly stable environments. And vice versa: detailed plans and "Big Design Up Front" (BDUF) work well in highly stable environment, but are potential sources of expensive rework for highly dynamic environments.
- ***Personnel***. Agile requires the continuous presence of experts capable of tailoring the processes to fit a precedented (Cockburn level 2[4]) or unprecedented (Cockburn level 3) new situation, while plan-driven methods can work with fewer experts of these levels once the initial project definition is completed.
- ***Culture***. Plan-driven methods "thrive on order", while Agile ones "thrive on chaos". The first thrive in a culture where people feel comfortable and empowered by having their roles defined by clear policies and procedures, the latter in a culture where people feel comfortable and empowered by having many degrees of freedom.

Most of the plan-driven processes provide explicit rules and guidelines for a range of situations, and they are meant to be tailored down (elements *removed*) to fit the situation at hand without producing too much overhead. Unfortunately, as noted in both Turner and Boehm (2003) and Kniberg and Skarin (2010), too often the full set of plans, specifications and standards are used, simply because the people implementing the process do not have the required expertise to remove unnecessary elements. This leads to needless work on account of the process itself, and ultimately to the perception of plan-driven processes as "heavy" and hampering development activities instead of supporting them.

[3]Adapted from Table 2-4, "The Five Critical Agility/Plan-Driven Factors", (2003a p. 117)

[4]The Cockburn levels are described in Appendix B.

Agile processes are often labeled "lightweight", as opposed to "heavyweight" traditional processes. This refers to the number and strength of constraints placed by the process on the developer. Agile processes partly rely on building up the process (elements *added* or *changed*) from a small set of rules, activities, roles and artifacts. True to their origins in Japanese industry, these usually include some form of continuous process improvement, enabling the users to add more elements as needed and/or remove those that do not work as intended. This tailoring requires a high level of process understanding within the team (level 3 and/or level 2 Cockburn experts) in order to work as intended.

The practice of developing software incrementally and iteratively (and by extension, Agile processes) grew from the increasing understanding of software development as inherently different from other fields of engineering. Brooks (1987) identified four inherent properties of software systems that largely explain the difficulties of development in general:

- **Complexity**. In software systems, no two parts are alike: "If they are, we make the two similar parts into a subroutine [...] In this respect, software systems differ profoundly from computers, buildings, or automobiles, where repeated elements abound." (ibid., p.2). Additionally, scaling up a software system implies increasing the number of elements and how they interact with each other, increasing the complexity more than linearly[5].
- **Conformity**. Software systems must conform to (interface with) existing institutions, systems and practices – sometimes because it is "the most recent arrival on the scene", sometimes because it is perceived as the part that is most easy to change in order to fit other parts.
- **Changeability**. "[...] the software product is embedded in a cultural matrix of applications, users, laws, and machine vehicles. These all change continually, and their changes inexorably force change upon the software product." (ibid., p.2)
- **Invisibility**. In addition to the complexity resulting from the number of different parts and the interaction between them, a software system also has a very large number of possible *states*. This makes conceptual modeling difficult at best. Any attempt to chart, model or visualize a software system will invariably capture only certain aspects or dimensions, as the whole is too complex for complete representation.

[5]A more thorough discussion of software complexity is found in Schneberger and McLean (2003), showing that complexity can be seen as resulting from 1) the number and variety of computing components, 2) the number and variety of component interactions, 3) the number and variety of component interdependencies, all affected by 4) the rate of change. ((Schneberger and McLean, 2003))

These properties make it extremely difficult to use an approach built on "getting it right the first time" – or "pure" waterfall – to design and build software systems. A natural way of handling complex problems is divide and conquer, the strategy of IID: build the system one small part at a time, and revisit and evolve parts developed earlier to improve them based on new information gained later in the process.

"Lean Development" is an Agile flavor of its own, but the roots of the Agile movement as a whole can be traced to "Lean" production and manufacturing. Lean owes much of its current popularity to the success of Toyota's product development processes from the 1980s onward, termed "lean production" in 1990 (Holweg, 2007). This approach[6] is characterized by its main goal: to maximize customer value while minimizing "waste" – the latter being anything (activities, processes, tools, roles etc. – literally anything) that does not add value to a product, value as perceived by the customer (Poppendieck and Poppendieck, 2003).

While it is beyond the scope of this thesis to investigate all the ways in which Lean has influenced Agile values and principles, a few key concepts can be seen to have an impact on UX-related issues in Agile development, and thus merit some attention here.

First; the Lean concept of maximizing *customer* value has been inherited by Agile directly. Most of the Agile values and principles (see Appendix A) can be said to be consequences of this overarching goal, and two mention the customer directly: the core value "Customer collaboration over contract negotiation" (Beck et al., 2001a) and the core principle "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software." (Beck et al., 2001b). This *customer*-centric perspective may be interpreted as conflicting with the *user*-centric perspective of User-Centered Design (UCD), as will be discussed in section 2.3.2.1.

Second, the Product Owner role in one of the most popular Agile flavors, Scrum, can be traced back to Toyota's "heavy-weight project manager" (Ballé and Ballé, 2005). In Toyota, the responsibilities and necessary skill sets of this role were well understood – among them the responsibility of concept creation and championing, and the ability to "forecast future customer expectations based on ambiguous and equivocal clues in the present market"

[6]The term "Lean" is usually combined with a postfix: Lean strategy, Lean principles, Lean way of thinking, Lean practices, Lean methodology, Lean philosophy etc. - in this context, the term "approach" was deemed sufficiently broad to denote the concept as a whole, across different disciplines.

(ibid., p.19). In Scrum, the responsibility of having a product vision and creating the concept is retained in the PO role. However, the knowledge, skills and competencies necessary in order to be able to both *create* a concept and to *communicate* it sufficiently well to the team is seldom elaborated or emphasized in Scrum literature aimed at practitioners. This may be seen as one of the root causes of why UX practitioners often end up in a facilitator role (discussed in section 7.1.6).

Third, the *time* and *productivity* factors have been heavily emphasized in the promotion of Lean ("doing more with less") – partly because the Toyota Production System made it possible to complete development projects in half the time of US equivalents, with four times their productivity (Ballé and Ballé, 2005). Toyota itself balances the stress from the work-intensifying effects of Lean with other practices designed to alleviate stress (Hampson, 1999) – and this is reflected in the eighth Agile principle, "Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely." However, the combined effect of Lean ideas of efficiency and the Agile focus on short timescales and early delivery of value – value often being interpreted as pure *functionality* – may lead to the dismissal of (time-consuming) UX-related activities and work as "waste" in certain settings.

2.1.2 Scrum

In this section, some central aspects of Scrum, mainly those relevant to understand terms used in the empirical study presented in chapter 5, are presented in a very brief and summarized form. This presentation is by no means meant to be exhaustive.

Scrum was codeveloped by Ken Schwaber and Jeff Sutherland in the early 1990s (Schwaber, 2004), and is currently in widespread use (see section 2.1). In the first paper it is presented as a development process that is "an enhancement of the commonly used iterative/incremental object-oriented development cycle" (Schwaber, 1995 p. 1). In the first book, it is presented as "a radically different approach to managing the systems development process" (ibid., p. 1) and "a management and control process that cuts through complexity to focus on building software that meets business needs" (ibid., p. 2), and in the latest version of the Scrum guide "a framework structured to support complex product development" (Schwaber and Sutherland, 2011 p. 5) Over the years, overwhelming amounts of Scrum-related literature have been published[7], but the description of the common rules, roles and artifacts still fit on the 16 pages of the official Scrum guide (ibid.).

[7]425 hits on "Scrum" in the Books section at Amazon.com at last search.

Scrum is based on empirical process control theory. The underlying idea is that software development is a complex process which is affected by many environmental variables that are continuously changing, and so the strategies chosen to control the process must be based on experience and what is known rather than the standardized, "repeatable and defined" process descriptions found in traditional models. Control is achieved by making the development transparent to those responsible for the outcome, frequent inspection of artifacts and progress, and adjustment of the process as soon as undesirable deviations are detected.

In the current Scrum version (ibid.), there are three main roles: the Product Owner (PO), (member of) the Development Team (DT), and the Scrum Master (SM). All are part of the Scrum Team. The PO is heavily involved in the process, and is responsible for managing the Product Backlog (list of product increments to be developed). The Development Team is self-organizing and cross-functional, and is responsible for developing the increments. The guide states that "Scrum recognizes no titles for Development Team members other than Developer, regardless of the work being performed by the person; there are no exceptions to this rule" (ibid., p. 6), but also that individual team members may have specialized skills and areas of focus. Accountability belongs to the Development Team as a whole. The Scrum Master ensures that the Team adheres to Scrum theory, practices, and rules, and serves both the PO, the Development Team and the organization in several ways.

Scrum prescribes four "events": the Sprint, the Sprint Planning Meeting, Daily Scrum, Sprint Review and Sprint Retrospective. The Sprint is a period of one month or less, in which a "potentially releasable" product increment is developed, and which contains all the other events. The Planning meeting is conducted at the start of each Sprint. In the first part, the PO presents prioritized backlog items to the team, and the team decides (forecasts) how much they will be able to complete during the Sprint. In the second part, the Development Team decides how the chosen work will get done. The Daily Scrum (often called the Daily or the Stand-up) is a short meeting of maximum 15 minutes that is conducted every day at the same time and place, usually in the morning. All DT members explain what they done since the last meeting, what will be done before the next, and what (if any) obstacles are in the way. The Sprint Review is held at the end of the Sprint. In this meeting, the work that has been completed during the Sprint is demonstrated to the PO. The Sprint Retrospective is usually held shortly after the Review. The purpose of this meeting is to inspect how the last Sprint went, and to identify what went well and possibilities for improvement.

It should be noted that although the central aspects of Scrum have mostly stayed the since the publication of the original paper in 1995, some changes to how it is presented have been made over the years, and many authors have provided additions and interpretations. Scrum rules and what it means to do Scrum "right" is continuously debated in the professional community.

2.1.3 *KANBAN*

In this section, a very brief overview of central principles and common practices in Kanban are presented. In order to gain an understanding of the process and how it may be implemented, two sources are recommended: the original book by Anderson (2010), and the comparison of Scrum and Kanban found in Kniberg and Skarin (2010).

Kanban is one of the newest additions to the Agile family of processes. The first kanban system used in software development was implemented in 2004, and the first book was published in 2010 (Anderson, 2010). The kanban concept is one of the many that can be related to Toyota and Lean manufacturing (see section 2.1.1.1). The word itself literally means "signal card", referring to the cards that are used to signal the need for restocking supplies at a point in a production line. Kanban systems are known as "pull" systems, as new work is pulled into the system when there is capacity to handle it, as opposed to "push" systems where work is pushed into the system based on demand. The advantage of pull systems is that the system cannot be overloaded, an advantage which resonates well with the Agile principle of sustainable pace (no. 8, Appendix A).

As described by Anderson, Kanban uses five core properties to "create an emergent set of Lean behaviors in organizations" (ibid., p. 15):

1. Visualize Workflow
2. Limit Work-in-Progress
3. Measure and Manage Flow
4. Make Process Policies Explicit
5. Use Models to Recognize Improvement Opportunities

The reasoning behind the importance of visualizing the workflow is that knowledge work like software development is inherently not visible. When it is visibly mapped out, it is easier to build an understanding of how the process works, and over time, to identify problems such as bottlenecks and make improvements. Also, the visualized workflow serves as a coordination tool, as the status of work items currently in the system as well as capacity use in the different phases are visible to all. Developers (and other team members) "pull" work from one phase in the workflow to the next as capacity becomes available.

Although the workflow may be visualized using various electronic systems, physical card walls are popular, mostly because they are continuously visible for all when placed at the premises of the team, and because they are very easily updated. The mapped-out workflow may start or stop at any point in the "value stream", that is, it does not need to be limited to the development process in itself. A set of columns are drawn, each representing a phase in the workflow, work flowing from left to right[8]. There are no explicit rules for exactly *how* a system should be visualized, as this needs to be tailored to every setting. A typical development workflow has some form of input queue, an analysis phase, development phase(s), test phase(s) and some form of "done" ("In production" or similar), all represented by a column.

Cards representing work items are moved across the card wall as work progresses. Cards may be designed in any way to fit the setting at hand, but a common design is using each card to represent a user story. Usually a code or ID is also found on the card, providing a simple way to locate additional information about the story in some electronic system. In some settings it is useful to define different types of work items ("User Story", "Bug", "Change Request" etc.), and to visualize the difference by using shapes and/or colors to signal the type. So-called "swim lanes" in the form of rows on the card wall may also be introduced, in order to cope with different types or sizes of work items.

A key strategy in Kanban to reduce lead time (simply put, the time it takes for a work item to move across the board) is reducing the amount of work being handled concurrently by the system. Anderson (ibid.) finds that there is causation between the quantity of work in progress and average lead time, and that the relationship is linear. The more work in progress at the same time, the more (calendar) time every item takes from start to finish. In practice, the strategy is implemented by limiting the number of cards allowed to be placed in each column. This limit is termed the "WIP limit" (work-in-progress limit), and is used on columns where needed. The actual number may vary. A common strategy is to use the number of people handling work in the column, plus 1-2 items.

There are no predefined time boxes or meetings in Kanban. In order to optimize the performance of the system, lead time is measured, and the process is continuously adjusted in order to lower the lead time. In order to be able to make the right adjustments, it is necessary to understand how the process works. Process policies are made explicit in order to make it possible to discuss process mechanisms based on a common understanding.

[8]Usually. Some use card walls where the flow runs from top to bottom.

2.2 User Experience (UX)

Over the past few decades, computerized systems have entered practically all areas of human activity. It has thus become increasingly important to design systems in order to be simple, effective and (in many cases) enjoyable to use. Interaction between humans and computers has been a subject field of study of its own since the early 1980s, and the field has grown ever since. The terminology related to the design of user-friendly systems has evolved with the field and the theories, methods and frameworks therein. The term "user experience" is among the broadest terms. It appears to have been introduced to regular use by Apple designers at a CHI (Computer-Human Interaction) conference in 1995 (Norman et al., 1995).

It is currently defined in ISO 9241-210:2010 as "A person's perceptions and responses that result from the use or anticipated use of a product, system or service.". Whether it is possible to *design* (all aspects of) *a* user experience is debated among professionals[9], but the goal of UX designers is to design *for* the best possible user experience. This requires taking more aspects into account than elements of the user interface.

A useful explanation is provided by the Nielsen Norman Group: "'User experience' encompasses all aspects of the end-user's interaction with the company, its services, and its products. The first requirement for an exemplary user experience is to meet the exact needs of the customer, without fuss or bother. Next comes simplicity and elegance that produce products that are a joy to own, a joy to use. True user experience goes far beyond giving customers what they say they want, or providing checklist features. In order to achieve high-quality user experience in a company's offerings there must be a seamless merging of the services of multiple disciplines, including engineering, marketing, graphical and industrial design, and interface design." (Nielsen Norman Group, 2010)

The majority of the literature within the field of Agile/UX integration discusses the combination of Agile processes and the coherent set of practices termed User-Centered Design (UCD), although use of the term UX seems to be increasing. This term was chosen for this thesis based on the fact that it is allows for a wider scope than the somewhat narrower concept of UCD, and avoids the confusion between practices and philosophies of "*User*-Centered Design" versus "*Usage*-Centered Design"(a comparison of the two may be found in Constantine and Lockwood (2002)). UX professionals (often titled interaction designers,

[9]http://aaronweyenberg.com/1934/why-im-not-a-ux-designer-and-neither-are-you

usability engineers, UX strategists etc.) work in a wide variety of ways, and the work may encompass anything from analyzing and planning business strategies to designing buttons for the user interface. As the field of Agile/UX integration is still young and lacks a commonly agreed upon, consistent terminology, it is the view of this author that using an all-encompassing term is conductive to a broader understanding.

2.3 UX and Agile

2.3.1 Is there a problem?

At first sight, there shouldn't be any problem – quite the opposite. As discussed by Nodder and Nielsen (2008), Agile holds the promise to address several issues that have long vexed UX professionals in traditional, "waterfall" processes. Short cycles mean less time between requirements specification and implementation, alleviating the issue of user needs changing before or during development. Direct communication and collaboration (partly) replacing written design documents should reduce misunderstandings, and increase the chance of developing what was *meant*, not what was *written*. Also, developing incrementally and iteratively should provide opportunities for incorporating continuous user research, testing and feedback, improving the user experience for every iteration.

Most research papers and experience reports are largely positive to the combination of Agile software processes and UX-related activities and processes. However, many have found that various Agile processes do not sufficiently address usability and user-centered design concerns (Constantine, 2002, Jokela and Abrahamsson, 2004, Blomkvist, 2005, Désilets, 2005, Chamberlain et al., 2006, Nodder and Nielsen, 2008) (the topic is further discussed section 3.2.1.1). UX practitioners experience several kinds of problems when trying to integrate their work practices and processes with Agile development processes:

- **Difficult to get the time to do overall, holistic design planning**
  Due to various reasons, "big design up front" (BDUF) is sought avoided, and this – somewhat mistakenly – has implications for interaction/interface design as well. (Further discussed below, and in sections 3.2.1.6 and 7.1.4.)
- **Difficult to prioritize UX-related features**
  The strong Agile focus on delivering working software frequently, and working software as the *primary* measure of progress, makes it difficult to prioritize resource-demanding, quality UX solutions above inferior solutions that require less time to develop.

- **Difficult to establish the UX role on the team**
  In the most widely adopted Agile flavors, XP and Scrum, no "designer" role is included, and so there are no guidelines as to how this role should work in relation to the development team.
- **Difficult to know when and how UX work should be performed in relation to development**
  Somewhat simplified, the customer (or in the case of Scrum, the Product Owner) is supposed to work with the team to provide requirements (user stories) to the extent they are needed, when they are needed. There are no guidelines as to when and how information for the user stories should be gathered, user research performed, detailed interface design and design iteration done, etc.

The annual AGILE conference in the US, organized by the Agile Alliance, is generally regarded as the "main" Agile conference – the number of attendees has ranged between 1 100 and 1 600 from 2007-2011[10], and the topics presented here both influence and reflect the interests of the community in general. Between 2003 and 2007, the number of accepted research papers and experience reports with an explicit focus on UX-related practices within an Agile context ranged between 0 and 3. In 2007 UX got its own track, and between 2008 and 2011 the number has ranged between 9 and 11.[11] The annual ACM CHI conference on Human Factors in Computing Systems may be seen as the UX equivalent to AGILE. In 2011, three courses on the topic were held: "Agile User Experience and UCD", "The Role of the UX Professional on an Agile Team" and "Experiencing Agile Usability: Breaking through the 'Us vs Them' Problem".

Although there is still little academic research on this topic compared to other Agile-related issues (technical, organizational, managerial), a body of work now exists that explores the topic both in a theoretical and an empirical perspective. In this section, some background information on basic issues is presented. A more detailed view is found in Chapter 3 (Related Work).

[10]The number of attendees having steadily increased from about 250 in 2003 ("Message from the Program Chairs", AGILE 2007 Conference Proceedings).

[11]Based on a manually performed search of the conference proceedings of AGILE from 2003-2011 (http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1000026). Research papers and experience reports that explicitly mentioned terms like 'usability', 'UX', 'user experience', 'user-centered design', 'UCD' or 'interaction design' in the title and/or abstract were included.

2.3.2 *Why is there a problem?*

As noted by several (Constantine, 2002, Jokela and Abrahamsson, 2004, Nodder and Nielsen, 2008), no representatives from the UX community were invited to participate in the formation of the Agile Alliance. The original 17 signees of the Agile Manifesto were software engineers – highly experienced, but still mainly concerned with the issues faced by *developers* (as in programmers/software engineers) in modern system development projects.

Also, the kind of projects that inspired Agile processes have certain key commonalities. Mostly, they were company in-house projects directed at developing products that would provide value by creating efficiency within the company, not by being sold to external parties. The users would thus not have much choice other than to use the products once developed. In that setting, it is difficult to prove the value of spending resources on UX.

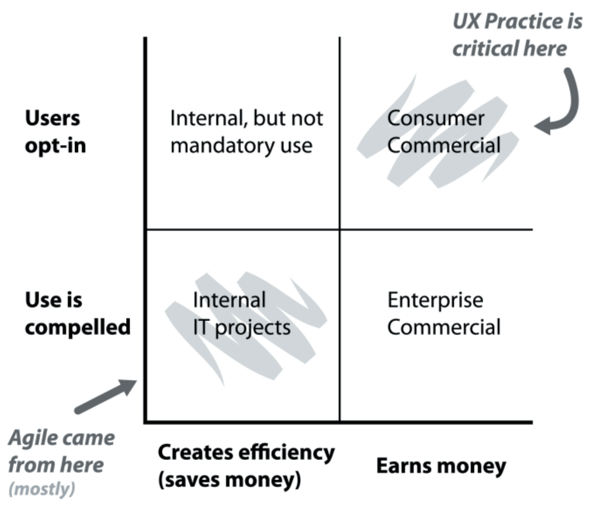

UX Practice is critical here
Users opt-in
Internal, but not mandatory use
Consumer Commercial
Use is compelled
Internal IT projects
Enterprise Commercial
Agile came from here (mostly)
Creates efficiency (saves money)
Earns money

Figure 2: Patton (2008b): characteristics of the products that inspired Agile vs. the products where UX Practice is critical

As shown in Patton's (2008b) diagram (figure 2), UX is more critical in settings where it provides business value more directly by making users prefer and buy products that provide better user experience than their competitors[12].

2.3.2.1 Customer-centered vs. user-centered

The key goal of Agile processes is customer satisfaction. The strong focus on the customer can be traced back to Toyota's successful production system, through Agile's roots in Lean

[12]Of course, better UX in products aimed at creating efficiency internally would probably create business value as well, increasing efficiency even more by making the system easier and more efficient in itself. However, this is often difficult to "sell" to internal decision makers who have to prioritize between more functionality, faster development time and better user experience.

(see section 2.1.1.1). However, where Toyota's customer is in fact the end user – those going to buy and drive the cars – such is not the case in many system development projects: often the customer is simply the representative of those paying the bill (the head of the department that has required the system, an external project manager, internal or external purchasing manager, etc.), rather than someone who will actually use the end product.

In Agile, good usability (insofar as it is considered at all) is thought to result from close and continuous collaboration with the customer. But as shown by Jokela and Abrahamsson (2004), this cooperation does not lead to good usability by itself. Based on the results of a usability process assessment of a controlled XP project, they argue that a considerable set of new activities should be added to XP in order to make the process "truly take usability issues seriously" (ibid., p. 12). However, they also found that the few usability-related activities that were conducted had a full impact on the design of the software, as these were conducted by the customer. Sy (2007) argues that UX specialists may be well-suited for the Agile customer role, as this might mitigate the customer vs. user bias.

2.3.2.2 Self-organization and overlapping skills vs. specialization

Team self-organization is an important concept in Agile, as stated in the eleventh principle: "The best architectures, requirements, and designs emerge from self-organizing teams"[13]. The concept can be briefly described in the words of Schwaber and Beedle, from the first Scrum book: "Where more traditional organizations call for the definition of static roles, Scrum developers can change hats dynamically, often passing as analysts, testers, coders, designers, architects and integrators in a single day. Simply said, everyone does everything that is in his or her power to deliver the system." (Schwaber and Beedle, 2002). The purpose of self-organization is to make the team as flexible as possible by enabling it to direct the resources to where they are needed, in contrast to "traditional" approaches where specific roles are only concerned with problems related to their areas of expertise.

Ideally, the team's shared responsibility and overlapping skill sets should make every task, topic and issue relevant for the whole team. However, this can be difficult to achieve in practice. In an ethnographic study of a professional Scrum team, Moe et al. (2008) found that highly specialized skills and corresponding division of work was the most important barrier for achieving self-organization. When the team members are constantly working on separate tasks, as will often be the case if there is a high level of specialization in the team, coordination activities such as daily meetings may be perceived as "waste of time" since the topics discussed are not (directly) relevant for everyone.

[13]http://agilemanifesto.org/principles.html

For a UX specialist working as a part of a developer team, this might lead to certain difficulties. Depending on the level of technical skill, it can be difficult for the specialist to participate in team activities such as estimating technical tasks and discussing technical issues. Also, discussing and/or estimating UX activities might not be perceived as relevant by more technically oriented team members. Organizing UX activities and personnel in a "parallel track" seems to be a general trend (see section 3.2.1.5), a trend that might partly result from this situation.

2.3.2.3 Scope level

Software processes are obviously not the only processes in an(y) organization, actual or defined. Software processes are surrounded by and intertwined with other processes at different levels. These processes differ by scope – what organizational activities the process covers. Pollice (2006) describes four levels of process scope: the personal process scope, the immediate team process scope, the project scope, and the enterprise process scope. As the process scope changes, the reason for having the process changes. At the personal and team levels, the purpose is to help developers provide a quality product, while at the project and enterprise levels the purpose is more strategic. The different processes need appropriate interfaces in order to work well as a whole.

As Hudson (2003) shows, Agile processes are focused on the *team* (introspective), while user/human-centered design processes focus on the user interface (extrospective). The UCD (UX) processes are related to the strategic levels (user experience as a competitive advantage) as well as to the project/product quality level. It might be said that Agile processes lack appropriate *interfaces* to UCD processes, partly on account of the mixed scope levels of the latter.

2.3.2.4 The Big Design Up Front (BDUF) problem

Very little time is devoted to an explicit, up-front design phase in Agile processes (Blomkvist, 2005). This is based on the notion that requirements will always change, and minimizing the time between requirements specification and development reduces the risk of developing functionality that is already obsolete by the time it is being developed. The user interface is shaped by iterative refinements throughout the process, and continuous collaboration with the customer is thought to provide the necessary input for creating a usable interface.

From a UX perspective, the lack of a design phase in which the product is considered in a holistic manner, taking all aspects affecting the user experience into consideration, may be considered a deficiency in Agile processes. Well-known UX professionals such as Alan Cooper have long advocated the view that there is a need for an up-front, overall planning phase considering the *interaction* design of the product as a whole, while detailed *interface* design may very well be left to the development phase (Cooper, 2008). Others have reasoned that it might all be a misunderstanding/miscommunication on account of different meanings of the term "design" (Ferreira et al., 2007b). This topic is further discussed in section 3.2.1.6 and 7.1.4.

3 RELATED WORK

In this chapter, related work in the field of Agile/UX integration is presented. Although there is still little academic research on this topic compared to other Agile-related issues (technical, organizational, managerial), a body of work now exists that explores the topic both in a theoretical and an empirical perspective. Section 3.1 describes the method used for identifying related work, and subsequent sections detail the key issues and general themes discussed in the reviewed material. An overview of relevant empirical studies may be found in section 3.2.2.

3.1 METHOD

Identification of related work was conducted in several stages. In the "pilot" stage, broad searches were performed using IEEE Xplore, Google Scholar and regular Google in order to identify common keywords, acronyms and terms used in the UX/Agile field. Articles, blog posts, forums and other sources found were read in order to gain an overall impression of recent progress in the field as well as current topics of interest in the professional community. In addition, conference proceedings from the AGILE and CHI conferences were manually searched. The relevant papers were recorded.

3.1.1 SYSTEMATIC SEARCH

In order to systematize the process of conducting the search in the second stage, the overall steps of conducting a literature review as described by Oates (2006 p. 80-90) were used, and some of the relevant 'lessons learned' in Brereton et al. (2007) were taken into account.

ACM Digital Library and IEEE Xplore were chosen as primary search engines. The search terms used were combinations of 1) keywords related to UX, and 2) keywords related to Agile (see table 1). The terms were combined using AND, as OR cast too wide a net and yielded mostly irrelevant results. In the first round of searching, abbreviations were included ("UX", "UI", "UCD", "HCI"), but it quickly turned out that all relevant papers would contain the full term ("User Experience", "User Interface" etc.). Since using the abbreviations only "polluted" the search results with irrelevant matches, these were omitted in the second round.

Also, the first round included "Extreme Programming" and "Dynamic Systems Development" as terms for the Agile concept. These were not included in the second round, as the resulting articles invariably also turned up when simply using the “Agile” term. However, “Scrum” and “Kanban” were still included in the second round, as these terms were more relevant and the search results should thus include all papers mentioning these in combination with UX.

In the second round, a total of twelve searches were performed for each search engine, using combinations of the following terms:

Concept	**Keywords**
UX	User Experience Usability User-Centered Design Human-Computer Interaction
Agile	Scrum Kanban Agile

Table 1: search terms used to identify related work

For each search, the title and abstract for each paper returned was read, and the papers deemed relevant were recorded. All duplicates were removed. The 24 searches returned a total of 306 papers, of which 135 were duplicates, and 74 were deemed relevant using the following criteria:

- must focus on the integration/combination of UX and Agile processes or practices (main or partial focus)
- must be peer reviewed
- must be available online and written in English

In this stage, only one literature review in the field turned up, and this did not appear to be exhaustive (Sohaib and Khan, 2010).

The third stage was added when an additional search in September 2011 found the systematic review by Silva et.al. (2011), published in August 2011. This review appeared exhaustive as well as qualitatively sound (evaluated according to the literature review criteria in Oates (2006)), and there was a very good match between the papers found in the second stage and the papers identified in the review. A comparison showed that the review included 15 papers that had not turned up in the ACM and IEEE searches, all of them from Scopus. These were included in the list of relevant papers.

The 89 papers were read and classified according to the categories used by Silva et.al. (2011): focus (design, evaluation or both), approach (generalist, specialist or generalist/specialist), and results (need for initiative, initiative proposal, lessons learned, and recommendations). Having read the complete papers, 15 were excluded on account of lack of relevance, and 9 papers were classified as "content duplicates" (papers written by one or more of the same authors as the original paper, but for a different audience/conference). However, 7 additional relevant papers were found through the bibliographies of some of the original papers.

24 papers reported on empirical studies of UX/Agile integration conducted by researchers. Of these, 9 were of special interest related to this thesis, and an overview and summaries of these are presented in section 3.2.2.

3.2 MAIN TOPICS/FINDINGS

3.2.1.1 Missing rules and guidelines for UX/Agile integration

Almost all the papers that focus mainly on UX/Agile integration discuss or mention the lack of rules and guidelines for integrating UX activities in Agile processes. In one of the most recent papers, it is stated that "Currently, there are no clear principles or guidelines for practitioners to achieve successful integration [...] None of the existing agile processes explicitly include principles and practices for understanding and eliciting testable and verifiable usability and user experience requirements." (Salah, 2011 p. 1). 17 papers are categorized as either "Need of Initiative" (concluding that there is a need for an approach for UX/Agile integration) or "Initiative Proposal" (suggesting an approach).

Düchting et al. (2007) present a systematic analysis of the user-centeredness of Scrum and XP from a theoretical perspective. As Agile processes lack the traditional phases like analysis, design, development and validation along with required deliverables and/or activities for each, system requirements are chosen as measurable criteria that provide a basis for the analysis. Three kinds of user-centered requirements are considered: usability requirements, workflow requirements and user interface requirements. A gap analysis is performed in order to evaluate elicitation and evaluation activities in Scrum and XP related to these requirements. The authors conclude that both Scrum and XP have "significant deficiencies" in handling user-centered requirements, and that these are treated insufficiently in all the important stages of development.

Several experience reports written by practitioners describe how the lack of UX-related focus in Agile initially created problems for the UX specialists/teams in the transition from "traditional", waterfall-like processes, and how these problems were overcome ("success stories"). Federoff and Courage (2009) mention that in 2006, when the organization started using Agile, "the vast majority of literature on the topic of agile did not include guidance on the inclusion of User-Centered Design (UCD) processes [...] The User Experience team was left with many unanswered questions" (p. 235). Budwig et al. (2009) initially encountered problems with concurrent design and development, communication and coordination issues as well as confusion about UX deliverables. In Cho (2009), the UX team initially faced some concerns and challenges related to roles, responsibilities, and merging the existing UX methodology in use (GDD) with the new Agile process.

Two papers mention that the lack of UX focus is not limited to Agile processes. Memmel et al. (2007 p. 167) state that "software engineering has to work with people with a background in HCI, but the course of collaboration is mostly unclear. It is therefore true that classic and agile SE methods "still lack explicit integration of HCI methods and processes" [...]". Constantine and Lockwood (2002) mention that interface design and usability are generally weak points in both heavyweight processes like the Unified Process and in lightweight processes like XP.

3.2.1.2 UX and Agile – a natural fit?

Despite the lack of UX focus in Agile, none of the papers deny the possibility of successful UX/Agile integration, and several focus on similarities of philosophy, activities and process that should make the two a very good combination.

Both the literature reviews found state that based on the reviewed material, UX/Agile integration can work well. The review conducted by Sohaib and Khan (2010) concludes that "Usability and agile are well compatible and they can work together" (p. V2-37). In Silva et al. (2011) it is found that according to all the experience reports identified, "Agile development and user-centered design are a natural fit." (p. 84).

Several papers written as a part of a research project on "agile software development methodologies with special emphasis on Extreme Programming and continuous usability evaluation", mostly by the same team of authors in varying combinations, are very positive to both the process and the results of UX/Agile integration. In an empirical study of how Agile User-Centered Design is carried out in practice by professionals (Hussain et al., 2009c), it is found that integrating usability/UCD into the Agile process has enhanced the quality and the usability of the product, and some participants point out that the integration has added value to their team and the process.

In (Hussain et al., 2009a), lessons learned from the combination of XP and UCD in a project are discussed, one of them being that “[...] the XP process fits well into the UCD approach because of the many overlapping principles [...]” (p. 178). In (Hussain et al., 2008), the necessity and benefits of including usability engineers and test users in an XP project are highlighted.

A survey conducted in 2009 with 92 respondents (Hussain et al. (2009b), described in section 3.2.2.1.4) concludes that it is easy to integrate usability/HCI techniques into Agile software development. However, it is not clear how the authors arrive at the conclusion that integration is *easy*, based on the questions and answers presented in the paper.

Two experience reports by different authors from the same organization also report mainly on very good results with Agile/UCD integration. From the papers, it is clear that UCD has a central role in product development in the organization, and that the careful adaptation of Agile to fit the organization's needs has taken this into account. In Honious and Clark (2006), it is stated that “Integrating UCD within the iterative cycle is not only possible but essential in creating a new customer-facing product that will win with users” (p. 1/pdf), and in Williams and Ferguson (2007) written one year later, the experience is that “As more development teams using UCD are finding, the iterative approach to agile is a natural fit for UCD.” (p.1/pdf).

A very interesting and unusual perspective is found in Patton (2005). This experience report describes how Agile was introduced as a way of solving the problems resulting from the time from requirements elicitation to working software being too long – unpredicted requirements and requirements would too frequently emerge when the customers saw the software for the first time. However, the new Agile requirements elicitation process creates new problems: too much time writing user stories, too many stories, difficulties prioritizing the stories and difficulties in finding dependencies between stories.

The author then introduced UCD techniques to solve these problems: collaborative elicitation and modeling sessions with domain experts/would-be users, on-site contextual observation, task modeling and prototyping. It was found that these techniques helped identify business goals, critical user constituencies and their goals, as well as understanding their workflow. User tasks were used in order to be better able to prioritize, estimate, and build a good candidate release plan. In essence: the integration of UCD helped “save” an Agile project that had become stuck.

3.2.1.3 Incremental vs. iterative

Many highlight the iterative/cyclical nature of both the Agile and the UCD process: UCD advocates iterative interface design, Agile builds working software in an iterative manner, and both build refinements on empirical information from previous iterations (Fox et al., 2008, Chamberlain et al., 2006, Hussain et al., 2009c, Memmel et al., 2007, Sohaib and Khan, 2010, Ferreira et al., 2007b, Najafi and Toyoshiba, 2008). However, as mentioned by Lee et al. (2009), the *incremental* nature of Agile development is somewhat at odds with traditional UCD. The resulting user interface can be disjoint, piecemeal and lack a cohesive overall vision. Problems related to this perspective are further discussed in section 7.1.5 and in the conclusion.

3.2.1.4 Short timeframes and UX as a bottleneck

The short timeframes of Agile development processes seem to be an issue that affects UX integration negatively in many cases. When functionality is to be developed within timeframes from 2-6 weeks, it is challenging to fit user research, interface design and user testing in.

Several experience reports by practitioners describe time-related issues. In Illmensee and Muff (2009), the organization's established UX practices were found to be too slow and cumbersome to be useful after the development department started using Agile. Research and detailed wireframes were thought to be too time-consuming, and did not seem directly correlated to the teams' primary coding efforts. In the beginning, UX was becoming a bottleneck, "and suddenly at risk of being marginalized" (ibid., p. 405).

After changing the UX practices to better match the new development processes, this was no longer a problem, but the authors have some concerns: "As iterations wore on and research continued, researchers began to feel the dull throb of fatigue. There was seldom time to rest or be still. In addition to design and development tasks, weekly research was extremely demanding physically and mentally. The pace was exhausting." (ibid., p. 409).

Before making considerable adjustments (most notably the "Office Hours" concept, rapid growth of the UX team, RITE testing and using the parallel track approach), the UX teams in Federoff (2008) and Leszek and Courage (2008) also experienced that there was not enough time to complete UX work within Agile timeframes. Before the adjustments were made, only 30% of UX team members responded favorably to the survey question "Is agile making your team more effective?". After the adjustments, the corresponding percentage was 73%, a considerable improvement.

In Williams and Ferguson (2007) , it was found that one UX person was not enough to maintain an active level of user engagement and still provide other deliverables related to the detailed design of the product on an Agile project. The solution was adding another UX person, creating two roles: the "UCD Researcher Role" and the "UCD Prototyper Role". The first would be responsible for user research and testing, while the other would create prototypes and write UI specifications.

Time issues are mentioned in research papers as well. The empirical study conducted by Kollmann et al. (2009) found that "insufficient UX team members is a big problem and could be a major source of frustration. Delivery has priority, so user research is the first thing to be compromised." (ibid., p. 17). In Chamberlain et al. (2006), the time problem experienced by the team is that the designers have a *shorter* iteration cycle than the developers. "The designers worked at a different pace to the developers, which made it hard to iterate around versions of software or designs" (ibid., p. 149). The implication that the developers had problems keeping up with the pace of the designers is somewhat at odds with the other findings in this section.

3.2.1.5 Parallel track

Variations on the "parallel track" approach to Agile/UX integration are described in several papers. Among the relevant papers, the earliest version is found in Beyer et al. (2004), followed by Miller (2005), although the most well-known appears to be the one by Sy (2007) (referenced by 13 of the papers)[14]. Additional versions are found in Budwig et al. (2009) and Silva et al. (2011). All mention that the approach should be combined with lightweight UX techniques, and that there should be extensive communication between designers and developers throughout the process.

For this approach, it is generally assumed that one or more UX specialists – people working mainly with UX-related tasks – are involved, working in parallel with the developers in a separate "track". The main idea is that up-front UX work like user research, requirements specification and design happens one or more Sprints/iterations/cycles before the developers start coding the functionality. During development, the UX specialists and developers discuss the tasks and give and receive feedback. One or more iterations after the functionality has been developed, the working code is tested and/or validated. Generally,

[14]Miller and Sy worked together in a UX section during the time period in which their company switched to Agile.

the first iteration in the project is used for up-front planning and doing user research – "Iteration 0".

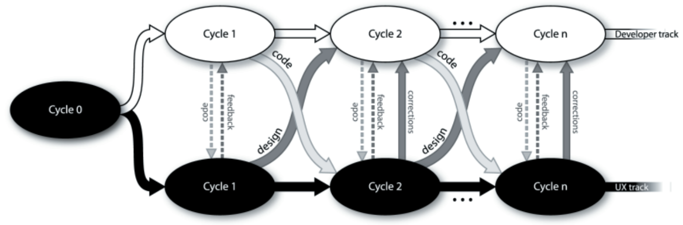


Figure 3: thumbnail of the parallel track model (full version may be found in Appendix C)

The version presented by Silva et al. (2011) is derived from the findings in the systematic review of papers on Agile/UX integration, and combines the most common practices and processes identified. A combination of the Silva et al. (ibid.) model and the Sy model (ibid.) is presented in Appendix C.

Sy (2007) mentions that prototypes were usability tested at least one cycle ahead of developers, to allow the User Experience team to iterate on designs. Contextual inquiry for workflows would be conducted at least two cycles ahead. She also discusses the problem of doing complex design work that would take more than one cycle (two to four weeks) to complete. The team solved this by introducing "design chunking": breaking large designs into small, cycle-sized pieces that incrementally add elements to the overall design. It is mentioned that since interaction designers are trained to consider experiences holistically, this may be difficult at first, but it is "a skill that comes with practice" (ibid., p. 120).

In the empirical study conducted by Fox et al. (2008), the researchers found that all the teams used the parallel track approach, and the approach is recommended in Kollmann et al. (2009), Budwig et al. (2009) and Federoff and Courage (2009). In Najafi and Toyoshiba (2008), a similar approach termed "staggered development method" was considered successful, although the UX team experienced that the focus on delivering functionality for every Sprint led to overall user goals being overlooked during development. The parallel track approach was also used in Fisher and Bankston (2009), but it was found that additional processes were needed in order to handle stakeholder alignment and to keep track of the work needed to make user stories ready for development.

3.2.1.6 Big/Little Design Up Front (BDUF/LDUF)

In a total of 13 of the papers found, the 2002 debate[15] between Alan Cooper (creator of the Goal Directed Design approach) and Kent Beck (creator of XP) is mentioned and/or

[15]The transcript is removed from both the original location and from the Web Archive, however a version may still be found at (Nelson, 2002). Note: in a discussion thread at ixda.org from 2009

discussed. Large parts of this debate focused on the issue of up-front design. While the interpretations in the different papers vary somewhat, the general view is that Cooper advocates all interaction design being done up-front and then handed off to the developers, while Beck advocates very little design done up-front. (In the view of this author, Cooper explicitly differentiated between *interaction* design – the process of "translating" user needs into a holistic, coherent product, which should be conducted closer to or as a part of an up-front requirements planning phase – and *interface* design, which can partly be left for subsequent phases and the programmers).

The lack of an overall, holistic plan regarding the features and interface of the product is often regarded as problematic. In their overview of common problems faced by UX practitioners working on Agile projects, Miller and Sy (2009) list "missing the 'Big Picture'" – the symptoms being no shared vision or end goal, too much focus on details, and hard to prioritize/make design decisions. In Obendorf and Finck (2008) it is stated that "Development 'feature by feature' can hinder a consistent design as the development process does not define the activity of designing a whole vision of the future system [...]", and in Meszaros and Aston (2006) that "Emergent Design doesn't work very well for user interfaces. Some Design Up Front seems to provide better guidance to the development team and provides earlier opportunities for feedback."

Most of the papers discussing this problem also propose techniques to alleviate or solve it, mainly involving some measure of design prior to development (Miller and Sy, 2009, Adikari et al., 2009, Obendorf and Finck, 2008, Meszaros and Aston, 2006, Detweiler, 2007, Kollmann et al., 2009, Hodgetts, 2005). Ungar (2008) finds that collaborative workshops held prior to development, focusing on design, foster shared understanding of design vision. Scenario-based design may also be used to ensure that an overall vision is retained during evolutionary development (Obendorf and Finck, 2008, Lee et al., 2011). Kollmann et al. (ibid., p. 17) also state that their findings support Cooper in the sense that "interaction design should be part of requirements planning and not limited to the delivery phase".

A very interesting perspective on this problem is found in Ferreira et al. (2007b) (findings described in section 3.2.2.1.7), in which use of the term "design" is discussed in relation to this problem. Work performed in order to evaluate business cases, appreciate the goals of the client, and to understand the work and mental models of the end users was seen by the participants as parts of what they regarded as *design*. "On the other hand, this kind of work might also be seen by developers as constituting *analysis* rather than design. Agile development advocates are wary of up-front design because it represents premature

(http://www.ixda.org/node/19673), Cooper stated that neither his nor Kent Beck's opinions are currently the same as they were at the time of the debate.

commitment, but if it is analysis and does not involve commitment, then it does not constitute the same kind of danger." (ibid., p. 15).

3.2.1.7 "Us vs. them"?

Some papers mention a division between designers and developers. In their empirical study of three teams, Chamberlain et al. (2006) (section 3.2.2.1.8) found that there seemed to be a fundamental problem of communication between the developers and the designers, and that there were power struggles between the disciplines. Some of the respondents in the study by Fox et al. (2008) mentioned that there was a barrier for the UX specialists to be accepted by the teams, and referred to this as the "us and them" perspective. Also, Miller and Sy (2009) list "poor communication" (between designers and developers) in their summary of the main problems experienced by UX practitioners on Agile projects.

The division is suggested attributed both to organizational and personal characteristics. Ferreira et al. (2010, 2011) (section 3.2.2.1.2/1) suggest that values and assumptions of decision-makers external to the teams shape UX/Agile practice. In one of the organizations in the study, UX and development divisions were separated, and there was no model or plan for coordinating the work. The authors find this to result from an underlying, organizational value: that the two disciplines should remain separate so that creative thinking will not be hindered by technical considerations. Kollmann et al. (2009) find that UX designers' understanding and attitude towards agile development affects their successful integration into the team.

3.2.1.8 Proposed methods and frameworks

Several proposed approaches to Agile/UX integration in various settings were found. Memmel et al. (2007) present CRUISER, an "agile cross-discipline user interface and software engineering lifecycle", which in many ways resembles the parallel track solution. It starts with a short up-front phase in which lightweight versions of classic HCI techniques (use cases, scenarios, prototypes) are used to describe user needs and task goals. In the second phase user interface and system architecture is developed in parallel, after a minimalist, common UI specification has been produced and the necessary interfaces identified. The construction and test phase closely resembles the equivalent phase in XP, but unit- and acceptance tests are used to ensure that UI-related issues are considered during development, and the authors recommend using "HCI personnel" during pair programming. They also recommend that the whole process should be "controlled by an authoritative person who must have a deep understanding of both SE and HCI" (ibid., p. 170).

Two approaches specifically target web-related development. In Benigni et al. (2010), USABAGILE_Web is presented as an approach for designing or redesigning a web-based system. This approach relies on the use of UX experts to perform evaluations of the user interface designs for every iteration. Constantine and Lockwood (2002) describe "Usage-Centered Engineering for Web Applications", a set of techniques targeted at novel web applications that "should integrate readily with lightweight or agile development processes under compressed development schedules" (ibid., p. 42).

The XPnUE approach proposed by Obendorf and Finck (2008) combines the Agile process XP with usability engineering techniques. In this approach, the usability-related weaknesses of XP are addressed by adopting techniques from Scenario-Based Engineering (SBE) and Rapid Contextual Design (RCD).

Beyer et al. (2004) present Rapid Contextual Design (Rapid CD), an approach that adapts classic CD to an Agile setting. The approach is based on four axioms: 1) separate design from engineering, 2) make the user the expert, 3) keep up-front planning to a minimum, and 4) work in quick iterations. The approach describes parallel tracks for designers and developers, and may be the first version of the "parallel track" model published.

3.2.1.9 Tools and techniques

Two examples of software tools developed to aid UX work on Agile projects were found. Hosseini-Khayat et al. (2010) present ActiveStory Enhanced, a tool for the creation and remote evaluation of low-fidelity prototypes. Hakim et al. (2003) present Sprint, a combination of method and tool designed to aid the coordination of specification assets. The tool links projects assets such as personas, scenarios, screen design, storyboards, requirements, and use cases, to form interactive specifications. It also keeps track of multiple asset fidelities, and eliminates asset redundancies.

Barksdale and McCrickard (2010) describe the use of Concept Mapping to facilitate effective collaboration and communication on cross-functional teams. Knowledge is represented through a collection of nodes and edges, where each node is a concept and each edge shows the relationship between concepts. The concept maps function as collaborative artifacts and negotiation tools, "facilitating collaboration while allowing team members' autonomy to operate flexibly in their area of specialty without imposing major constraints on the other" (ibid., p. 4692).

Several have found that user testing and design iteration may be speeded up while still providing the necessary feedback. An example is the "Rapid Experimentation" approach described in Meckem and Carlson (2010), which employs iterative and quick testing of an idea using low-fidelity simulation, small research teams and hypotheses that are very narrow in scope. It resembles the RITE method mentioned in Patton (2008a), a form of usability testing that involves fixing the problems discovered by one test subject before performing another test, instead of performing several tests and combining the results. Patton advocates RITE testing on paper prototypes as well as on working software.

A more process-oriented technique that appears to be useful in settings where UX resources are scarce, is the “Office Hours” concept described by Leszek and Courage (2008). UX specialists dedicate given times and days each week to provide UX-related support, so-called “Office Hours”. These times are registered in public Outlook calendars, and the development teams may sign themselves up for any available slot when needed.

3.2.2 Empirical studies of Agile/UX integration

10 papers were found that fulfilled the following criteria:

- main focus Agile/UX integration and/or some aspect of integration
- empirical study conducted by researchers
- study conducted in a “real world” setting and/or involving practitioners

These papers were found to be particularly relevant for this thesis, as the method and/or context make the results somewhat comparable to those in the empirical study (chapter 5). The results are summarized in this section. An overview of key properties of each study is given in Table 2. No contradictory results were found between the studies, although the perspective and focus varied.

Study	Method	Participants/ projects	Agile process	Context	Purpose
Ferreira et.al. (2011)	Case study (one team/10-day observation period, 3 semi-structured interviews)	14 developers and 2 UX designers in one project/company	Scrum	Internet division in a large media organization (U.K.)	Better understanding of Agile and UX in practice
Ferreira et.al. (2010)	Case study (two teams)	1 team of 14 developers + 2 UX designers (same data as for Ferreira et.al. (2011)), 1 team of 5 developers and 1 UX designer	Scrum	Internet division in a large media organization (U.K.)/ Small organization developing for mobile devices (U.K.)	Study how the settings in which Agile developers and UX designers work together shape their work
Kollmann et.al. (2009)	Qualitative approach (10 in-depth interviews of UX practitioners, 2 days of observation)	10 UX practitioners	Scrum / combination of Scrum and XP	Varying (large/small organizations, 1 freelancer)	Study the role of UX practitioners on agile projects, as perceived by the UX practitioners themselves
Hussain et.al. (2009b)	Grounded theory (semi-structured interviews, some observation/ document analysis)	5 interviewees from different organizations	Scrum/ combination of Scrum and XP	Varying	Investigate how the integration of agile methods and user-centered design is carried out in practice
Hussain et.al. (2009c)	Online survey	92 respondents	Various Agile	Varying	Investigate the current state of the integration of agile methods and usability/UCD
Fox et.al. (2008)	Qualitative study (10 in-depth interviews)	10 participants from different teams, all with some degree of formal or informal UCD training	Not specified (general term Agile is used)	Varying	Investigate how agile methods are currently being integrate with UCD practices, validation of previous work by Sy (2007)
Brown et.al. (2008)	Ethnographic study	2 designers and 1	Not specified	Company developing	Study collaboration between interaction

		developers (1 team/project)	(general term Agile is used)	educational computer games (Canada)	designers and software developers, with an emphasis on the role of artifacts/show how this can be studied
Ferreira et.al. (2007a) and (2007b)	Qualitative study (8 interviews)	8 interviewees: 1 UX person and 1 developer from a total of 4 teams	XP (3 teams), Scrum (1 team)	Varying	Study how UI design fits into the structure of agile iterations
Chamberlain et.al. (2006)	Ethnographic/ qualitative (14 observation sessions, 10 interviews)	3 project teams within the same organization	Scrum, (2 teams), XP (1 team)	Large media company	Identify and investigate the issues faced by project teams trying to integrate UCD and agile development

Table 2: empirical studies of Agile/UX integration

3.2.2.1.1 Ferreira et al. (2011)

Ferreira et al. (2011) report on an observational study of a mature Scrum team in a large organization, and how this team interacts with the UX designers working on the same project. The day-to-day activities of Scrum were observed for the duration of one Sprint (10 work days), and the sprint planning meeting, the daily stand-ups, the retrospective and the "ad hoc" meetings between developers and UX designers were attended. During the initial period, interviews were conducted with the developer roles interfacing most frequently with the UX designers. After this period, an interview was conducted with one of the UX designers to gain more insight into the UX perspective on issues that had previously emerged.

In this organization, the UX and development divisions were separated both organizationally and physically. There was no model or plan for how the Agile development and UX divisions ought to coordinate their work, and the authors claim that this separation was supported and sustained by the values and assumptions within the organization about how developers and designers will best produce quality software. UI designs were produced by the UX division and then "handed off" to the development division for implementation. The developers would then start by performing a "gap analysis", identifying mismatches between the design and the software already implemented. When such were found, the developers' realization prompted further action to obtain a more detailed explanation from the UX designers.

Developers spent significant amounts of time becoming familiar with UX designs that were sent to them. The UX designs could not be directly implemented, as parts had already been implemented in previous Sprints, and the gap analysis revealed mismatches in other parts. One developer was responsible for systematically analyzing the design and writing story cards, which were then prioritized by the team. The UX designers were approached for design direction when unanticipated implementation issues arose. The developers and designers made decisions only within their disciplines, and the developers were not comfortable creating their own UX designs, as this had led to waste of work when the UX designers suggested changes later.

The authors discuss the observations using concepts from the area of Computer Supported Cooperative Work (CSCW). UX design and Agile development has elements of *cooperative* work, as it is performed asynchronously (where *collaborative* work would be synchronous). However, while cooperative work assumes mutual dependency between the actors, in this case it appeared that only the developers experienced a dependency on the UX designers, not the other way around.

Based on the study, the authors suggest implications for practice that involve culture, self-organization and purposeful work, focusing on three aspects:

1. Consider work (sub)culture. Supporting Agile developers and UX designers "requires an understanding of the realities of what constitutes work for each group and managing the expectations of cooperation and collaboration" (ibid., p. 972).
2. Enable self-organization. This requires an organizational setting that promotes and reinforces self-organizing behavior, instead of impeding it.
3. Value purposeful work. Interaction between Agile developers and UX designers is localized, contingent and purposeful work, and should be supported and valued.

3.2.2.1.2 Ferreira et al. (2010)

In Ferreira et al. (2010), the same team as in the previous section ("Team1") is presented alongside another team in order to demonstrate how the practices of the two teams are shaped by the values and assumptions of decision-makers external to the teams. The other team – Team2 – is part of a small organization developing solely for mobile phones, and is composed of five Agile developers and one UX designer. The UX designer and Product owner are seated separately from the developers (different floors).

In the case of Team2, the authors could not observe any distinct and visible UX/Agile integration activities. All roles on the project were "taking part in frequent discussions, creating awareness and sharing decision-making responsibilities" (ibid., p. 181). Both the developers and the designer were comfortable discussing, carding and prioritizing all types of work, including UX and non-UX-work. Every role was involved in UX design discussions, leding to decisions that considered both design values and technical constraints. The managing director informed the authors that he hired individuals who could contribute outside their roles as well as fulfill their own.

The underlying, organizational values in the two cases are stated as follows. Value1/Team1: Agile and UX disciplines should remain separate. The theory behind this value is that the designers should be free to use their "creative energies" unencumbered by technical issues. Value2/Team2: Agile and UX disciplines should work as closely as possible. This value hinges on the assumption that each role has a valuable contribution to make to the overall product.

3.2.2.1.3 Kollmann et al. (2009)

The study by Kollmann et al. (2009) focuses on the role of UX practitioners on agile projects, as perceived by UX practitioners themselves. Ten UX practitioners from various settings (self-employed freelancers, part of in-house teams, working at digital agencies or UX consultancies) were interviewed, and the work of a designer on an agile team in a large organization was observed for two days. The authors identified two main themes that are perceived by UX practitioners to be highly influential in the success of integrating UCD and Agile approaches: the UX practitioners' understanding of their job role (identity), and the need to establish, protect and communicate an overall team vision.

The study finds that the identity of UX practitioners on agile projects is “shaped by their understanding of agile and their attitudes towards flexibility and change” (ibid., p. 14). A good understanding of Agile as well as active participation in the process is perceived to facilitate the practitioner's ability to work in an agile context. The participants saw Agile as a philosophy rather than a framework or toolkit, and while several had faced obstacles or were not entirely satisfied, none were negative about agile. “Rather, they believed that UCD has to be flexible and adapt to it into agile – making it more agile in itself.” (ibid., p. 14). The UX job role is found to be influenced by the structure and practices of the organization, and by the understanding others have of that role and responsibilities, a view that is reflected in both of the studies by Ferreira et al. (2010, 2011) described in the previous sections.

Another main theme found is that of creating and maintaining a “UX vision”, the envisioning of the experience the user should have when using a product. One of the biggest disadvantages of agile is thought to be the risk of losing the UX vision in the process, as time pressure and lack of UX resources makes it difficult to include user research as well as establishing and communicating the vision. Also, UX designers recognize the need for a shared team vision that balances stakeholders' goals including UX, development and business concerns.

Based on the study results combined with previous work by others, the authors conclude with four general recommendations for UX/Agile integration. First; UX practitioners should become design facilitators, participating in generating and prioritizing features and user stories. Second, UX work should be broken down into manageable chunks and done in a parallel track that feeds into development (as discussed in section 3.2.1.5). Third, including end users can be done successfully by strategies like using a user panel or scheduling user testing no matter what. Fourth, lightweight tools like personas and scenario-based approaches make UCD more agile.

3.2.2.1.4 Hussain et al. (2009c) and (2009b)

Hussain et al. (2009c) investigate how the integration of Agile and User-Centered Design (UCD) is carried out in practice, by using a Grounded Theory approach. Five professionals (usability specialists, generalists and program/project managers) from different contexts are interviewed, and the data is supplemented with some observation and document analysis.

The authors find that almost all results are consistent with previous findings by others (the studies described in sections 3.2.2.1.5 and 3.2.2.1.7 are mentioned). There is an increasing realization of the importance of usability in software development, providing advantages for the integration of usability/UCD activities in Agile processes. All participants mention that integrating usability/UCD has enhanced the quality and the usability of the product, and increased the satisfaction of the users. There is mutual understanding of each other's work between UCD professionals and developers, and mutual learning.

The same authors report on the results of an online survey with 92 respondents in Hussain et al. (2009b). To the authors' knowledge, there were only two other surveys mainly addressing Agile/UX integration at the time of writing: the one by the Nielsen Norman Group (Nodder and Nielsen, 2008), and the one conducted by the practitioner Anders Ramsay (Ramsay, 2010). The survey was targeted at practitioners (both UX persons and developers) working with Agile processes and integrating some HCI techniques, and was conducted online. Respondents were recruited through online special interest groups (UX groups and Agile groups) and personal networks. The survey received responses from all over the world, although most were from Europe and North America (46% and 38% respectively). Respondents were mainly UX persons (36%), followed by developers (17%), the rest being managerial/consultant/other. 51% had 2-5 years experience with agile, followed by 23% with 1 year of experience, the rest having from 6-20 years(17%) and some did not answer the question (9%).

Most respondents answer “Strongly Agree” or “Agree” (29%/43%) when asked to evaluate the statement “The integration of agile methods with usability/UCD has added value to the adopted process and to the teams”. Similar percentages are found when reviewing answers to “The adoption of an agile UCD process has resulted in the improvement of usability and quality of the product developed” (21% strongly agree, 45% agree) and “The resulting product has increased the satisfaction of its end-users due to the adoption of an agile UCD process” (22% strongly agree, 41% agree). Interestingly, all three questions have a rather high rate of “Don't know/no answer” (15%, 15% and 24%, respectively).

The authors conclude that “Agile software development methods are flexible, iterative, and lightweight, making it easy to integrate usability/HCI techniques into them, while the focus of both methodologies on delivering value and on customers/users, as well as their iterative nature and continuous testing, further facilitate and enable this integration [...]. The survey results support this as the majority of respondents perceive that the integration of agile methods with usability/user-centered design has added value to their adopted process and to their teams. They also perceive that the adoption of an agile user-centered design process by their teams has resulted in the improvement of usability and quality of the product developed and has also increased the satisfaction of its end-users." (Hussain et al., 2009b p. 424). As mentioned in section 3.2.1.2, it is not entirely clear how the authors reach the conclusion that integration is "easy", based on the aforementioned questions/answers.

3.2.2.1.5 Fox et al. (2008)

The paper by Fox et al. (2008) reports on a qualitative study using the Grounded Theory approach. In-depth, semi-structured interviews were conducted with ten participants of varied backgrounds and roles from Canada, the U.S. and Europe in order to investigate Agile/UX integration, and to validate the work of Sy (2007). The data was analyzed using open, axial and selective coding, and a set of overall concepts and themes were uncovered. The authors note that while none of the processes described by the participants were identical, the approaches they used had some commonalities, and these were used to construct an overall picture.

It was found that all teams followed processes comparable to the "parallel track" model described by Sy (2007)[16] (see section 3.2.1.5), and the authors state that "This confirms the model [...] and suggests that this integration approach is widely applicable" (Fox et al., 2008 p. 71). However, differences in terms of team roles and responsibilities were found. The paper describes three different approaches regarding who performs the UCD-oriented activities in a team: the Specialist, the Generalist and the Specialist-Generalist approaches.

In the *Specialist* approach, the team uses specialists for UX work. In the *Generalist* approach, only two roles are used: the users/customers and the developers acting also as UX specialists (based on informal or self-taught UX expertise). The Generalist Specialist approach is a kind of hybrid, in that the person(s) performing UX work is both a UX specialist as well as a software developer. The authors mention that the data suggest that the working environment in the Generalist's approach was much less formal than that of the environment of the Specialists.

All but two of the participants suggested that integrating UCD practices into their Agile process had added value to their development process. All of the approaches did have at least one UX person (the term User-Centered Design Specialist, UCDS is used) or a team

[16]The authors present the parallel track model as *originally* presented in Sy (2007). However, in the literature reviewed for this thesis it appears that this approach was first described/recommended in Beyer (2004).

member acting as it. Additionally, it was found that in all cases, some upfront effort was invested by all participants, although the design effort was rather limited compared to traditional processes. On average, the participants spent four weeks on the initial stage (pre-coding stage).

3.2.2.1.6 Brown et al. (2008)

Brown et al. (2008) use cultural-historical psychology as a theoretical framework for analyzing the role of artifacts in interaction and collaboration between interaction designers and software developers, a main point of the paper being to show how this collaboration can be studied. A video recording of two designers and one developer is submitted to a highly detailed analysis using five complementary analysis techniques. The paper/analysis is part of a larger study aiming to better understand software team collaboration, and the UX/Agile focus was chosen "because this kind of collaboration does not feature strongly in agile development." (ibid., p. 49).

The study results suggested a coherent set of interpretations. The key role of artifacts in the collaborative process was confirmed, and it was found that the collaborative meeting progressed in identifiable phases. The phases primarily alternated between creating and reflecting phases, and these were distinguishable by the number of artifacts (lists of flaws and good aspects, whiteboard sketches and design stories) introduced in the phases. The artifacts were shown to extend psychological processes such as remembering, self-regulating, enhancing focus and making associations. Sketches and stories were used in combination because this helped to reveal errors in a design, and helped to show how the user would use the software. The analysis suggested that artifacts in combination are more powerful.

3.2.2.1.7 Ferreira et al. (2007a) and (2007b)

Two papers by Ferreira et al. (2007a, 2007b) report on the same study. The first was presented at the XP 2007 conference, the other at the Agile 2007 conference. The study was a qualitative study using the Grounded Theory approach, based on interviews with team members from four projects at different companies, each in different countries (the U.S., Ireland, New Zealand and Finland). Two of the teams were developing web-based software. For each team, someone concentrating on UX and someone concentrating on development was interviewed. The primary objective was to understand the process and practices relating to user interaction design on XP software projects.

The study focused on UI design in Agile development. In the first paper (2007a), the first emergent theme is that iteration planning affects UI design. In three of the teams, the iteration planning determined what elements would next be developed as functional software, thus affecting which parts of the UI will be elaborated next. One team used a different approach, as it was their goal to complete the UI design before any development began, and then partition the design into parts and work with the developers to schedule implementation of the parts. Interestingly, this delivery schedule seemed to result in a similar sequence of work.

In the second paper (2007b), one of the results is that the participants were clear about the advantages of doing interaction design before implementation begins, as it helped mitigate risks, helped designers achieve "the best possible" design, while contributing to cost and time savings by ensuring better project estimation and prioritization. However, most participants believed that although some up-front design was essential, the UI should not be completely specified up-front, as some room needed to be left for decisions to be made during implementation. Participants also emphasized the close collaboration that interaction designers had with the business/marketing segment and clients and end-users before development began.

Time spent doing up-front interaction design was not regarded as a problem by the participants, as they saw the agile warning against up-front design as pertaining to *code* design, not UI design. There was an understanding that issues of cost and time are actually the *reasons* why the UI design should be iterated using prototypes rather than programming, as prototyping is more cost effective. "Most important, there is evidence of understanding that the issue is not whether interaction design should be done up-front or not, but rather when up-front interaction design will reduce risk, and is therefore advisable, rather than increase risk by making premature design commitments." (ibid., p. 16).

Another set of findings was that development iterations drive usability testing, and that usability testing results in changes in development. The working software produced from each iteration was seen as an opportunity to perform user testing earlier than would have been possible otherwise (using traditional processes), and following iterations were seen as opportunities to change the software to accommodate the results of the testing.

3.2.2.1.8 Chamberlain et al. (2006)

In the study by Chamberlain et al. (2006), three project teams (each with both developers and designers) in a large media company with strong UCD traditions were observed for around 2-4 hours per week on site by one person for a period of 6 months. Additionally, ten interviews with people of different roles were carried out in order to gain further insight into the observations. The approach was ethnographic in nature, as the researcher had no a priori hypotheses to test.

The authors found that there seemed to be a fundamental problem of communication between the developers and designers within each team. They discuss the issue of power struggles, mentioning that part of the reason why the UCD and Agile methodologies came about is that each discipline needed a defense mechanism against other disciplines (management/business) taking away their power. One of their conclusions is that power struggles may be overcome if there is some balancing role or mechanism put in place to ensure that each discipline has equal power on the team.

All the teams used prototyping: one used an evolutionary approach, another a throw-away approach, and the third used a combination of both. Two of the teams experienced time problems with prototyping: one had little time to handle prototyping effectively, resulting in the cycle of prototyping and feedback not working as planned. On the other team, the designers had a shorter iteration cycle than the developers, and the authors mention that this may ultimately have contributed to the abandonment of Scrum by the designers on this team.

The authors conclude that UCD and Agile methods are compatible, but that they can also be at odds due to power struggles between developers and designers, that development usually takes more time than creating (UI design) prototypes, communication issues if members of the team don't take part in some elements/phase of the project, a reluctance to understand the needs of other project elements, and the extent to which the user is able/willing to contribute to the project. It is also stated that these problems can be overcome by having a power balancing mechanism, good resource/project management, having all team members involved at key points in the project, and ensuring that the (end) user is involved in the project in order to ensure that the end-product will work in a realistic situation.

3.2.2.1.9 Summary of empirical study findings

Study	**Main findings**
Ferreira et al. (2011, 2010)	Two teams in separate organizations were studied, and the authors found that • Agile/UX integration practice is shaped by values and assumptions of decision-makers external to the teams • supporting Agile developers and UX designers requires an understanding of what constitutes work for each group, and managing the expectations

	of cooperation and collaboration • organizations should enable self-organization, and support interaction between developers and designers
Kollmann et al. (2009)	10 interviews of practitioners and some observation was conducted, and it was found that • two main issues are relevant for practitioners designing the UX in an Agile context: identity (role) and vision • the identity of UX practitioners on Agile projects is shaped by their understanding of Agile and their attitudes towards flexibility and change • creating and maintaining a (holistic) vision of what the product should be is important. The vision is a *mutual* understanding of what the product should be, and what the goals are from a technical, a business, and a UX perspective Four implications for practice are suggested, see section 3.2.2.1.3.
Hussain et al. (2009b, 2009c)	In (2009b) five practitioners were interviewed, and the authors found that • there is an increasing realization of the importance of usability in software development • participants find that integrating UX into their Agile process has improved product quality, and added value to the process and the teams • some up-front work is generally carried out to understand users, their goals and the project vision • designers and developers appreciate each other's work, and each group can learn from the other In (2009c), a survey of 92 practitioners largely confirms the results from (2009b). The authors conclude that integrating UX into Agile processes is easy, although it is not clear how this conclusion is reached based on the presented questions from the study.
Fox et al. (2008)	Ten practitioners were interviewed, and the authors found that • all teams followed processes comparable to the parallel track model • all teams spent some time doing up-front work related to UX, four weeks on average • most teams found that integrating UCD had added value to their

	development process Also, three approaches to team roles and responsibilities regarding UX were identified: the Specialist, the Generalist and the Generalist Specialist approach. All approaches had at least one designer or a member acting as a designer.
Brown et al. (2008)	One collaborative meeting of two designers and one developer was analyzed in detail, and it was found that • artifacts (lists, sketches, user stories) have a key role in the collaborative process • artifacts in combination are more powerful
Ferreira et al. (2007a, 2007 b)	8 practitioners, two from each of four teams were interviewed, and the authors found that • there are advantages to up-front interaction design. Most, but not all, interaction design should be done up front • iteration planning affects UI design • development iterations drive usability testing • usability testing results in changes in development • agile changes the relationship between UI designers and software developers
Chamberlain et al. (2006)	Three project teams in one organization were studied, and the authors found that • UCD and Agile methods are compatible, but can also be at odds due to o power struggles between designers and developers o time differences: development usually takes more time than creating (UI) prototypes o communication issues o reluctance to understand the needs of each element of the project o the extent to which the user is able/willing to contribute to the project • these problems may be overcome by implementing countermeasures, see section 3.2.2.1.8

4 METHOD

In this chapter, available research paradigms, methods and how these have been used in the field of Agile/UX are described, the chosen methods and techniques for the study are discussed, and the research design presented along with a description of how the study was conducted.

4.1 RESEARCH PARADIGMS AND METHODS AVAILABLE

4.1.1 POSITIVISM AND INTERPRETIVISM

There are a number of different research approaches available for researchers studying different aspects of Information Systems (IS) and computing. As discussed by Oates (2006), most research within this field is based on one of three underlying philosophical paradigms: positivism, interpretivism and critical research. Positivism is the oldest, and for many it is considered the only “proper” kind of research (ibid., p. 283). This paradigm underlies the well-known Scientific Method, which has two basic assumptions: that the world is ordered and regular, and that we can investigate it objectively. The aim is to find universal laws, patterns and regularities.

According to Oates (ibid.), the shared worldview of researchers working within the positivist paradigm has certain characteristics. It is assumed that the world exists independently of humans, that is, there is a physical and social world that can be studied, captured and measured in an objective way, the researcher being a neutral, objective, and impartial observer. Research is conducted by testing hypotheses empirically, leading to confirmation or refutation. There is a strong preference for mathematical modeling and proofs, and statistical analysis. Quantitative research methods are often (but by no means exclusively) used within this paradigm.

Finding the laws, patterns and regularities is mainly done by *experiment*. The scientific method has three basic techniques: reductionism, repeatability and refutation. Complex things are broken down into smaller things that are more easily studied. Experiments are repeated in order to ensure that results are not somehow affected by faulty equipment, measurements or similar. Experiments are also (ideally) constructed with the aim of refuting hypotheses, not confirming them, as we can never prove something is true for all time: just because it hasn't happened/been observed before, doesn't mean it will never happen/be observed.

However, while the scientific method and the positivist paradigm has been very successful in studies of the natural world, it does have limitations when studying the social world and patterns of behavior. Interpretivism provides an alternative: “Interpretive research in IS and computing is concerned with understanding the social context of an information system: the social processes by which it is developed and construed by people and through which it influences, and is influenced by, its social setting.” (ibid., p. 292). As opposed to the positivist belief that the world “out there” can be studied objectively, the interpretivist worldview is characterized among other things by the belief that there are multiple subjective realities – different groups or cultures perceive the world differently – and meaning is dynamic and socially constructed.

The aim of the research is to understand people and their actions in their worlds, not in the artificial world of a laboratory. It is not expected that one will arrive at one fixed explanation of what occurs in the study. Instead, multiple explanations are offered and discussed, specifically in terms of which has the strongest evidence. While quantitative methods can be (and are) used within this paradigm, qualitative methods are strongly associated with it.

Importantly, within the interpretivist paradigm researchers are *not* assumed to be neutral: “Their own assumptions, beliefs, values and actions will inevitably shape the research process [...]” (ibid., p. 293). Researchers must therefore acknowledge how they influence the research (self-reflectiveness). One of the quality criteria for judging positivist research is objectivity, but this is not appropriate when operating within the interpretivist paradigm, in which it is believed that there will always be bias: “[...] no observation can be made independently of how a researcher chooses to conceptualize them on the basis of prior theory (developed by people) or previous experiences.” (ibid., p. 293).

4.1.2 Research methods available, and methods used to study Agile/UX integration

There are a large number of research methods available, and most of the main methods have been used to study UX/Agile integration or some aspect of it. Yin (2009) discusses the suitability of different methods judged from the form of the research question, whether the method requires control of behavioral events and whether the research focuses on contemporary events.

Experiments may be suitable for studying contemporary events in situations in which the researcher(s) has control of behavioral elements (as for instance in a laboratory), and the research questions are on the “how” or “why” forms. The only study of UX/Agile integration classified by Silva et al. (2011) as experimental was Jokela and Abrahamsson (2004), in which it was studied to what extent the Agile process XP supported development of usable software. In this study, the researchers set up an XP project that was designed to follow the principles, rules and guidelines found in XP literature, using students as team members. Two additional studies on this form were found during the literature review (Barksdale and McCrickard, 2010, Humayoun et al., 2011), both using students to test Concept Mapping and Automated Tool Support.

Surveys may be used in order to answer questions on the “who, what, where, how many, how much” forms, do not require control over behavioral events, and focus on contemporary events. The main idea is that the same kinds of data will be obtained from a large group of people (or events) in a standardized and systematic way. The data are then analyzed in order to look for patterns and associations. Although surveys are usually associated with questionnaires, data can also be gathered using interviews, observations and documents (Oates 2006). To this author's knowledge, only one survey on Agile/UX integration published in peer-reviewed journals exists to date, described in section 3.2.2.1.4. Another survey was undertaken as part of the background work for the report by Nodder and Nielsen (2008), and a third conducted by the practitioner Anders Ramsay was found in the general search[17].

Yin (2009) mentions Archival Analysis and History as methods for studying non-contemporary events. Berg (2004) uses the term “Historiography” to denote “a method for discovering, from records and accounts, what happened during some past period” (ibid., p. 233). When studying Agile processes, many emphasize *change* compared to previous processes, and use written records and oral histories to investigate how things were done before in order to be able to compare. In “internet time”, historical events might not be that far in the past.

The case study research method is described in more detail in section 4.1.3. This method has been used to study Agile/UX integration previously, as in the studies conducted by Ferreira et.al. (Ferreira et al., 2010, 2011) and Barksdale and McCrickard (2010). Several papers written by practitioners use the term case study rather than experience report (Illmensee and Muff, 2009, Budwig et al., 2009, Najafi and Toyoshiba, 2008, Miller, 2005).

Action research, in which the researcher is actively involved in the events/concepts studied and which aims at practical outcomes (Oates 2006), is also represented in the papers reviewed. In Lee et al. (2009, 2011), this method was used to develop and implement an

[17] Main findings are presented at
http://www.andersramsay.com/2010/11/06/findings-from-the-state-of-agile-ux-survey

approach known as eXtreme Scenario-based Design (XSBD), as well as study the results of the implementation in a project in a (commercial) organization.

Ethnography originated in anthropology, and is mainly concerned with studying people, cultures and sub-cultures. The aim of an ethnography might be to produce a rich, detailed picture of events or cultures, and "[...] is said to be successful if its readers are able to understand the activities of people in another culture (or sub-culture) and see that they make sense *within the context of that culture.*" (Oates 2006, p. 179). Brown et.al. (2008) used an ethnographic approach in order to study collaboration between user interaction designers and developers. Chamberlain et al. (2006) also mention that "The initial approach to observation was ethnographic in nature in that the researcher approached the activity as 'strange' and had no a priori hypotheses to test."

Another research method in the Information Systems and Computing field discussed by Oates (2006) is "Design and Creation". This strategy "[...] focuses on developing new IT products, also called *artefacts*." (ibid., p. 108). Artifacts can be working systems, but also methods/methodologies, models and constructs (like the notions of entities, objects or data flows). Several researchers have used this strategy in the papers reviewed. Hosseini-Khayat et al. (2010) describes the system "ActiveStory Enhanced", a tool for creation and remote evaluation of low-fidelity prototypes, aimed at easing the incorporation of usability evaluation into Agile software development practices. Hakim et al. (2003) present Sprint, "[...] both a method and a tool designed to allow an Agile approach to product development while supporting best practices for user-centered design." (ibid., p. 1).

In several papers, the researchers state that they use a grounded theory approach (Ferreira et al., 2007a, Ferreira et al., 2007b, Brown et al., 2008, Winter et al., 2011). This approach is particular to qualitative research, where the intention is to do field research and then analyze the data to see what theory emerges (which would then be grounded in the field data). Oates (ibid.) mentions that there are several versions of it, and that researchers using it should be aware of (and state) which version they are using. Although only one of the papers mention the version explicitly, based on the method literature referenced it would seem that most use the Strauss and Corbin version, as described by Coleman and O'Connor (2007). In this version, the research questions are stated before the study is undertaken, and it is assumed that the researcher enters the field with some knowledge of the phenomenon (including knowledge of relevant literature in the field of study).

4.1.3 TYPES OF CASE STUDIES

Yin (2009) presents a twofold, technical definition of case studies:

"1. A case study is an empirical inquiry that

- investigates a contemporary phenomenon in depth and within its real-life context, especially when
- the boundaries between phenomenon and context are not clearly evident.

2. The case study inquiry

- copes with the technically distinctive situation in which there will be many more variables of interest than data points, and as one result
- relies on multiple sources of evidence, with data needing to converge in a triangulating fashion, and as another result
- benefits from the prior development of theoretical propositions to guide data collection and analysis." (ibid., p. 18)

Three basic types of case studies are described in Yin (2003): exploratory, descriptive and explanatory. When the existing knowledge on a topic is scarce, and there are no conceptual frameworks of hypotheses of note in the field, an exploratory study might be undertaken. This kind of case study is often used in initial phases of larger studies, in which the exploratory study is used to help the researcher(s) define questions or hypotheses for later phases/subsequent studies. Although this kind of study is likely to occur before theories or specific research questions are formulated, it should be preceded by statements about what is to be explored, the purpose of the exploration, and the criteria by which the exploration will be judged successful (Yin 2009).

Descriptive case studies are aimed at providing a rich and detailed analysis of a particular phenomenon and its context, discussing what occurred and how different people perceive what occurred (Oates 2006, p. 143). Explanatory studies go further, and try to explain *why* the observed outcome happened. Yin (2009) describes several techniques for explanation building in case studies. He states that the explanation-building process is likely to be iterative. An initial statement is made, compared to the findings of an initial case, and the statement is revised. Other details of the case are compared against the revision, and the revision is compared to the facts of a second, third or more cases.

4.1.4 Research methods and techniques chosen

While the general area of interest was clear from the beginning – working with user experience within Agile processes – an extensive period of literature research was needed in order to establish an understanding of the current status in the field. The search process and the results are described in section 3 (Related Work), with some additional background information described in section 2 (Background).

4.1.4.1 Choosing the Case Study approach

The decision to use the case study method for the investigation in this thesis was based on the definition of a case study by Yin (2009) presented in section 4.1.3, combined with his conditions for which situations this method is relevant. The case study is a relevant method in situations where a) the form of the research question(s) posed is "how" or "why", b) the investigator does not have control over behavioral events, and 3) the investigation focuses on contemporary events (ibid., p. 8).

It might be argued that Agile/UX integration could be studied through experiments, as called for by Silva et al. (2011). However, while experiments might yield interesting results, it is the view of the author of this thesis that this phenomenon has to be studied in its real-life context in order to be better understood. There are simply too many conditions in the context that can severely affect the integration (as described in the Background section, and in section 3.2.2, Empirical studies of Agile/UX integration): the type of company, type of product, managerial values and preconceptions, experience with Agile, implementation of Agile, beliefs and personality of the team members, software processes used previously, customer expectations etc – the boundaries between the integration and the context within which it occurs, are indeed not clear.

As is usually the case with research designs, this one has undergone some changes during the research process. In the initial design, an online survey was envisaged in addition to the case studies. The survey was designed and tested on several UX practitioners as well as developers working in Agile settings. However, during testing it became clear that while the survey might yield some interesting statistics (for instance, how many UX resources there usually are to a team), the key point studied – the integration itself – was simply too complex to describe and ask about within an online survey that would be filled out by volunteer professionals. An example is questions about the "parallel track" solution – as the term is not generally recognized by professionals, although they might be using the solution itself, it required explanation (adding validity issues) as well as invariably producing the answer *it depends* from all test subjects. Also, experience as well as feedback indicated that open-ended questions would be too time-consuming for the subjects to fill out, combined with yielding too short/brief answers to give sufficient insight. After some redesigns and tests this method was abandoned, as it was decided that it would not yield sufficiently valid and usable results for the study as a whole.

As previously mentioned, several of the empirical studies on Agile/UX integration use a "grounded theory" approach, albeit the version in which research questions are stated prior to the study, and assumes some prior knowledge of the field on the part of the researcher(s). In those studies, the purpose has largely been explorative, and the focus is directed toward "emerging themes". In this study, those themes are among the sources for the development of the theoretical propositions used. Yin (2009) argues that for case studies, theory development as part of the design phase is essential, although the "theory" in question should "by no means be considered with the formality of grand theory in social science [...] Rather, the simple goal is to have a sufficient blueprint for your study, and this requires theoretical propositions [...]" (ibid., p. 36).

4.1.4.2 Choosing interviews as main data collection technique

Interviews are used as either the main or as a complementary data collection technique in almost all of the empirical studies reviewed in section 3. As mentioned by several (Oates 2006, Yin 2009, Berg 2004), this technique is much used in case studies (and ethnographies). Interviewing can be defined as a conversation with a purpose, where the purpose is to gather information (Berg 2004, p. 75). This technique can be suitable when the researcher wants to obtain detailed information, ask complex questions that might need different wording or logic for different people, explore concepts (like experiences) that cannot easily be observed or described via pre-defined questionnaire responses, or investigate sensitive issues or privileged information that respondents might not be willing to give in written form to a researcher they have not met (Oates 2006, p. 187).

Agile/UX integration is a form of human activity, like software development itself. While the activity can be studied through observation, understanding the motivations and logic behind the actions requires talking to the subjects (actants) themselves. A problem with this approach is of course that the results might not reflect reality: the interviewee(s) could have a faulty recollection of events, could deliberately answer incorrectly, could be influenced by the interviewer etc. Combining interviews with observations could to some extent have helped overcome such problems, but gaining access to development teams for (especially) long-term observation is difficult, and the observation itself could have affected the behavior observed. Additionally, the time and resources available did not allow for a multiple-case study if observation was to be included, and the information gained from multiple cases conducted mainly by interview was deemed more valuable (in view of the research questions) than the information from one observational study would have been.

As discussed by Berg (2004), at least three major categories of interviews may be identified: standardized, semistandardized and unstandardized. These can be organized along a continuum of formality of structure, where the first is most formally structured, the second "more or less structured" and the third completely unstructured. All of the interviews conducted in this study fall in the semistandardized category. Semistandardized interviews may be characterized by the following: questions may be reordered during the interview, wording of questions is flexible, level of language may be adjusted, interviewer may answer questions and make clarifications, and interviewer may add or delete probes to interview between subsequent subjects.

The semistandardized form was chosen because although the concept studied was the same across all cases, considerable flexibility was needed in order to be able to gain enough information about context variables as well as personal opinions and perceptions. As the field of UX/Agile integration lacks a consistent terminology and conceptual framework, practitioners may use different terms for the same (or similar) concepts, and they might follow certain patterns without being aware that it *is* a pattern. A completely unstructured approach would not have been time effective, as the research design was based on certain propositions and central themes were known in advance.

4.2 Research design

The purpose of the study is mainly descriptive, although some limited attempts are made at explaining certain observed phenomena. The study follows a multiple-case, embedded design, with five limited case studies. Empirical data have been collected mainly through interviews, although (very) limited observation and documents/data from issue trackers have been used as well. Analysis of the empirical data follows one of the four general strategies described by Yin (2009) – relying on theoretical propositions. The propositions are derived/inferred from the literature review conducted prior to the empirical study, and are stated below.

4.2.1 Research questions

1. How can UX-related work practices and processes be integrated with Agile software processes?
2. How are UX-related work practices and processes integrated with Agile software processes in practice [in a web-related environment]?

4.2.2 PROPOSITIONS

I) As Agile software development processes are mainly developer- and code-oriented, it requires conscious and focused effort to effectively integrate UX-related work practices and processes.

II) A common pattern of integration is the one called “parallel track”, as described by Silva da Silva et.al. (2011). This form of integration would be expected in cases where there is a) one or more dedicated UX specialists on the team, and/or b) where the Agile process in use is relying on specific time boxes (e.g., “Sprints” in Scrum) for development, and/or c) where UX has an established, valued position in the development organization.

III) UX tasks and activities are usually recorded, discussed, estimated and tracked separately from technical tasks.

IV) The amount of up-front design work done in an integrated environment varies. For products that are more information-heavy than interaction-heavy, more up-front work can be considered necessary.

V) The short time frames in Agile software processes make it difficult to perform enough user testing on actual end users.

VI) The UX specialist role will often evolve into a facilitator role in the project, coordinating customer-oriented activities as well as “driving” parts of the development effort.

VII) UX professionals generally work well with developers on Agile projects, particularly when they have a technical background and/or understanding.

4.2.3 UNIT(S) OF ANALYSIS

The main unit of analysis is the *integration* of UX-related work practices and processes and Agile software processes. This implies the following sub-units of analysis for each case:

- UX-related work practices and processes that are used
- the Agile software process used

- the product(s) and project(s) involved in each case
- the UX professional(s) or developer(s) mainly responsible for the UX-related work practices and processes
- other main stakeholders of the integration (developers, Product Owners/Customers, Scrum Masters, Project Managers) where applicable

4.2.4 SELECTING THE CASES

The five cases were selected based on the following rationale:

- two cases were chosen in order to study settings in which the "parallel track" pattern is likely to occur: one or more full-time UX resources combined with teams following the Scrum development process.
- two cases were chosen in order to study UX integration in teams following the Kanban development process.
- one case was chosen in order to study integration in a setting with very limited UX resources.

All the cases provide data for all the propositions (cross-case synthesis). However, the "parallel track"-cases are especially relevant for proposition II).

In the initial research design, it was planned to study mainly "parallel track" settings with different levels of UX resources available. However, during the study the opportunity arose to study integration in teams using the Kanban process. As this process is very new in a software development context (first book published 2011), this was considered a very valuable opportunity, and two of the originally planned cases were replaced with the Kanban cases.

4.3 HOW THE STUDY WAS CONDUCTED

Details about how each separate case study was conducted are described in chapter 5. The overall process was started by contacting two experienced UX professionals whom the author knew to be very interested in the subject of UX/Agile integration, as they had both held public talks on the subject. Both agreed to do preliminary interviews, and these were held shortly after. During the interviews, the subject was explored on general terms, and it became clear that there is currently a lot of interest regarding the subject within the professional community. This impression was further corroborated by several blogs, community sites and professional forums on the web, in addition to the information gathered during the process of identifying related work.

For each of the interviews, an interview guide was written. All of the guides were more or less similar in structure. In the first part of the interview, questions mainly related to the subject and the relevant organization. Usually the questions would then center on the project and the product being developed, followed by questions about the development process currently in use. All subjects were also asked about UX work practices and processes, although this part was naturally limited for some of the interviewees, like the developers. In cases A and D, where there were multiple interviewees, separate guides were prepared for each subject in order to capture aspects of the processes from several perspectives.

During the actual interviews, the interviewees would be encouraged to speak as freely as possible, in order to minimize the influence of the interviewer on the content. In most of the interviews, few detailed questions from the guide actually had to be asked, as the interviewees would generally provide very long and detailed answers to the overall "could you describe..." and "how..." questions. For every interview, permission was asked to use an audio recorder, and this was always given. At the end of the interview, the open question "is there anything else you want to tell me?" was asked to ensure that the interviewee was given opportunity to cover all aspects he or she thought were important. After each interview, information about the time, date, place and how long the interview lasted, were noted.

In all of the cases, additional information was retrieved from the relevant organization's web site. For several projects, project-specific information and presentations were publicly available on the site. In the cases where the interviews were conducted on the premises of the projects, the author was able to make observations of artifacts, seating arrangements and similar, and in cases A and D meetings were observed as well. For cases C, D and E, additional documents were provided by the interviewees. All audio files, transcripts, notes, photographs and other documents were gathered in separate case files.

4.4 CONTENT ANALYSIS

A broad definition of content analysis is found in Berg, citing Holsti (1968): "[content analysis is] any technique for making inferences by systematically and objectively identifying special characteristics of a message." (Berg 2004, p. 267). Berg explains how objectivity can be accomplished by means of explicit *criteria of selection*: the researcher shows how inclusion or exclusion of content is done consistently according to these criteria, so that other researchers (or readers), when looking at the same data, would arrive at the same or comparable results.

The content analysis follows one of the four general strategies for analyzing case study evidence described by Yin (2009), relying on theoretical propositions. The study was designed according to a set of theoretical propositions (stated in section 4.2.2) derived from the literature review (described in chapter 3), and the data collection was guided by these. During the interviews, the external line of inquiry consisted of open-ended questions and probing, all the while being guided by the internal line of inquiry based on the propositions.

The data collected from each case was then analyzed separately, by looking for information relevant to the propositions in each unit (interview transcript, interview/observation notes and other documents). In addition to looking for factual information and corroborating this with information in the documents obtained in each case (information about the organization, project and product), a set of specific questions were answered for each transcript:

- how does the described development process differ from the standard process model? Are changes and/or reasons for changes mentioned?
- how and when is UX research usually performed, if ever? [in relation to the development process]
- how and when is UX design work usually performed? [in relation to the development process]
- how and when is UX testing usually performed? [in relation to the development process]
- how are UX tasks and activities usually recorded/discussed/estimated/tracked?
- how does the subject describe his/hers or the UX specialist's role?
- how is the relationship between developers and UX specialists described (if applicable)?
- how is UX work perceived in the organization?

In order to help ensure that no evidence relevant to the propositions was overlooked, the transcripts were coded according to a scheme developed over several iterations. In the initial iteration, a code was used for each proposition (deduction), but it was quickly discovered that several codes would usually overlap, and that the analysis would rely too much on subjective interpretation in the coding process. In the second iteration, a scheme was developed according to a set of themes that appeared to be prevalent in most of the transcripts (induction), but this strategy eventually produced an overly complicated and detailed set of codes. In the final iteration, the coding scheme was based on a simplified combination, arranged into a loose hierarchical framework, presented underneath.

Main theme category	**Code**	**Subcategory 1**	**Code suffix**	**Subcategory 2**	**Code suffix**
People	#ppl	Interviewee (subject)	_subj		
		Team	_team		
		Stakeholders	_stake		
Process	#proc	UX	_UX	The role of the UX specialist	_role
				User research/elicitation of user needs	_need
				Designing user interface and/or information architecture	_des
				User testing (any)	_test
				UX terminology	_term
		Development	_dev		
		Project	_proj		
		Organization	_org		
Product	#prod				

Table 3: final coding scheme used for coding the interview transcripts

5 Results

5.1 Introduction

In this chapter, the results from five limited case studies are presented. The main data source of every case has been between 1-5 in-depth interviews, although other sources have been used as well. All of the interviews were conducted within a time frame of three months in the fall of 2011, and took place in the city of Oslo, Norway. Every interview lasted from a minimum of about an hour, to a maximum of three hours. All interviews were conducted in Norwegian, the native language of both the interviewees and the author. With the exception of three of the preliminary interviews, all were recorded, transcribed verbatim and coded according to the scheme presented in section 4.4.

While the presentation of the results is of course kept as close to the original data as possible, the translation – especially regarding the quotations – has been somewhat challenging. In addition to all of the regular challenges of translating from one language to another (which is of course a subject field all of its own), it appears that a certain specialized vocabulary of trade has evolved within the web-related professions. Many words and expressions are to a certain extent derived from English, but some terms are regular Norwegian words used in a specialized sense to signify some role, practice or artifact that is commonly used within the profession.

An example would be the Norwegian word "skisse", which usually means something akin to the English word "sketch". When used in its specialized meaning, it very often signifies a highly detailed mockup of the user interface, often crafted to be pixel-perfect (though not always), although it *might* also be used in the regular sense – a quickly drawn, wireframe-like picture. This word has mostly been translated as "mockup", although "wireframe" has been used when it was clear from the context that the interviewee meant the less detailed variety. Another is the use of "functional", derived from "functionality", in terms like "functional team" (team in which the members mainly work with requirements gathering and specification, usually without developers) and "he's a func" (as in "he is one of the persons who mainly work with requirements gathering and specification, not a developer").

In the quotations, the translations are sought to be held as close to the original wording as possible, and the colloquial style preserved. The following markings have been used:

- **[...]** is used to mark that words or sentences were removed, not being relevant
- **..** (two dots) mark a short pause
- **...** (three dots) mark a slightly longer pause

- **[]** words in brackets have been inserted or replaced by the author

A code is inserted after each quotation. The first part is the participant code; the second part refers to the line number in the transcript. While the complete transcripts are not included, some relevant information might still be gained from this (like the relative placement of one quotation to another).

5.1.1 Overview

The following table presents a set of key characteristics for each case. It should be noted that both the number of Agile teams and the number of UX specialists have varied throughout the projects in all cases. This is further detailed within each case description.

Case	Process	Organization type	Number of Agile teams on the project	Number of UX specialists dedicated to the project
A	Scrum	Enterprise in the private sector, several thousand employees	2	2 (one full time, one part time)
B	Scrum	Department in the public sector, several hundred employees	11	3
C	Kanban	Department in the public sector, several thousand employees	1	2 (one full time, one part time)
D	Kanban	Enterprise in the private sector, several thousand employees	1	1
E	General Agile	Organization in the public sector, several thousand employees	N/A	1 (time divided between projects)

Table 4: overview of the cases

The following table presents the individual interviews in the order they were conducted. All interviews lasted between one and three hours.

Interview no.	Participant code	Agile process (current)	Years of Agile experience	Working as	Case
1	P1	Scrum	6	Interaction Designer	Preliminary interview
2	P2/C-1	Kanban	5	Interaction Designer	Preliminary interview
3	P3/E-0	General Agile	*[18]	Section leader (middle manager)	Preliminary interview
4	E-1	General Agile	*[19]	Interaction Designer	E
5	C-1	Kanban	5	Interaction Designer	C
6	D-1	Kanban	6	Project leader	D/Preliminary interview
7	D-2	Kanban	2	Product Owner	D
8	D-3	Kanban	2	Developer	D
9	D-4	Kanban	3	Interaction Designer	D
10	D-1	Kanban	6	Project leader	D
11	B-1	Scrum	6	Interaction Designer	B
12	A-1	Scrum	4	Project leader	A
13	A-2	Scrum	1	Usability Specialist	A
14	A-3	Scrum	11	Developer	A
15	A-4	Scrum	6	Interaction Designer	A
16	A-5	Scrum	5	Product Owner	A

Table 5: overview of the interviews conducted

[18]This rather depends on the definition of Agile. The process described in Case E has evolved from a more "traditional" variety into an ever more Agile one (as defined by Abrahamsson et al. (2002), see section 2.1.1) over the last 6-7 years.

[19]Same as for E-0

5.2 Case A

5.2.1 How the study was conducted

A usability specialist (A-2) was contacted by e-mail and asked for an interview. He responded favorably, and provided several additional contacts on the same project. Interviews were scheduled with A-1, A-2 and A-4 by e-mail. A-3 and A-5 were contacted and interviews scheduled on site. All interviews were performed during a two-week period, and took place at the premises of the organization. Seating arrangements, Scrum boards and various artifacts were studied at the premises. All interviews were recorded, shortly after transcribed verbatim, and coded according to the scheme presented in section 4.4. Some additional facts and figures were retrieved from the organization's web site.

5.2.2 Context

5.2.2.1 The organization

The organization is a large international enterprise, with several thousand employees in Norway. A Usability group of eight people resides in the Customer Experience department, but additional UX resources are hired to work on various projects. Currently there are six UX people working on web-related projects, and the work is coordinated by one of the members of the Usability group (A-2).

According to the interviewees, the main process models used in the organization are based on stage-gate, "waterfall"-like models. The first Agile-based (Scrum) project was started in 2004 (the project described in this case), but large parts of the organization still use the traditional models for development projects. This is seen as somewhat challenging by the interviewees handling communication and coordination between projects.

5.2.2.2 The project(s)/product(s)

The project is part of a large project originally started in 2004, aimed at creating "the best customer web in the world". The project (parts of the main project) has been stopped twice, once because a new round of requirements gathering was needed, the second time mainly because there were not enough people on the technical side to handle two large sub-projects at the same time.

For a time, the project was Norway's largest Scrum project with 7-8 teams working in parallel, and a total of approximately 200 people working on the project. The number of teams has varied during the project. At the time of writing, there had only been one team for some time, but very recently the team was scaled up and split in two as a result of increased demands.

For several years, the project was focused on the task of creating a complete, integrated self-service web for the customers. The PO (A-5) was one of the main initiators of this part, having a background from the Customer Services department. The goal was to make it possible for the customers to do any task related to the administration of their subscriptions and products directly on the web, instead of having to call customer service. From a technical perspective, this has been challenging, as it has involved integrating several large subsystems in a single sign-on solution.

A large part of the project consists of developing the customer interfaces for ordering products (like subscriptions) on the web site. As new products are continuously being developed in the organization, new requirements have been constantly fed into the project. The formal project is in its last phases, but the teams will continue design and development work in the foreseeable future, while maintenance and operations will be outsourced.

5.2.2.3 The interviewees

A-1 took over as Project Leader a few months ago, hired from a consulting firm. He has about five years of experience with Agile development processes, and has the equivalent of a master's degree in Information Technology.

A-2 has been working as a usability specialist in the organization for about a year. A few months ago he was given the role of UX lead for web, and is now responsible for coordinating UX-related work across all web-related projects. Until recently he spent most of his time on another, closely related project, but is currently involved in both. He has a master's degree in Industrial Design, and has about a year of experience with Agile development processes.

A-3 has been working as a developer on the project for five years, hired from a consulting firm. He finished his (equivalent to) master's degree in Computer Science in 1996, and has eleven years of experience with Agile processes.

A-4 is also a hired consultant, has been part of the project for three and a half years, and was for a time – until recently – the only UX person on the team(s). He finished his master's degree in HCI and Interaction Design in 2007.

A-5 is the Product Owner of the team, and has been employed by the organization since 1999. This is his first project as a PO. He became a certified PO some years ago, and has a total of five years' experience with agile development processes. He has a varied university education, and is currently working on his master's degree.

5.2.3 Relevant findings

5.2.3.1 The development process

For several years, the project employed a certified Scrum coach as a project leader. Although the process mostly follows Scrum "by the book", some significant add-ons and adaptations have been made, among them two pre-planning meetings every Sprint and a mandatory QA session for every finished story *before* Sprint Review.

The (currently) two teams run two-week sprints, with joint Daily stand-ups, Sprint Plannings and Sprint Reviews, but separate pre-planning meetings. The whole project team is seated in the same room, with the interaction designers sitting right next to the developers. Everyone attends Daily stand-up, including the interaction designers and the PO. The number of team members has varied, but currently 10-12 people report at Daily.

The Sprint starts every Monday with Sprint Planning. Every Wednesday, a pre-planning meeting for the upcoming Sprint is held. All team members, the interaction designer(s), the project leader and the PO attend. For the first meeting, "pre-plan 1", the PO brings the user stories from the backlog that currently have the highest priority. He presents the stories, and the developers (and others) ask questions about them. If possible, the questions are answered directly during the meeting. If additional work or clarifications are needed, the questions are left to be answered at the next pre-planning meeting ("pre-plan 2").

The developers break the user stories down into tasks, which are written both on post-it notes and in the issue-tracker (Jira). During this procedure, additional questions or information needs may surface. Developers may assign tasks to others. For instance, if a mockup is needed, a task titled "create mockup" is assigned to the interaction designer. As the system interacts with several subsystems, issues may arise in this area, and these are generally handled by a separate systems architect. The whole team is responsible for doing anything needed to make the stories ready for development.

All the new and updated information is then reviewed in pre-plan 2. When all the information needed for developing a story is present, the story gets a "readyreadyready" status and is put in the "Ready" queue. If a story is "a long way from ready" in pre-plan 1, it may not even get to pre-plan 2, but is held for the next pre-plan 1 or a subsequent one, whenever enough information is gathered. Stories may sometimes loop the pre-plan meetings several times before they get "readyreadyready" status.

During Sprint Planning, the developers decide which and how many tasks they are able to take on for the sprint. The total number of user stories developed per Sprint varies, as the stories may consist of a different number of tasks. Development velocity is tracked using story points. User stories should always be small enough to be developed during one Sprint, but sometimes technical issues make it necessary to split one over two Sprints or more. Velocity points are given only when a story is done, the team does not receive credit for stories that are "partly done".

The pre-planning meetings were instated in order to avoid stalling caused by missing information and/or clarifications:

> *"Yeah, it was because the stories weren't ready when we got them. And we saw that it costs the team a lot to start working on stories that aren't ready. When we start on a story that isn't ready, then.. if it's impeded when it's almost done, so that it's stalled for two-three sprints without being finished, you've lost almost everything when you start again."* [A3-467]
>
> *"Because if we don't [have them], impediments may arise.. and noise, like "how is this supposed to work, and how is that supposed to work... and then you lose focus."* [A1-913]

The stories and development tasks are tracked on a physical Scrum board. When the team was recently split in two, each team got its own board, and the two are placed side by side. The columns are "Ready", "Not started", "In progress" and "Done". There are also two additional kinds of status markers that may be attached to a story or task, "Impeded" or "QA". The Daily stand-up is conducted in front of the boards. For a time, the team tried using only the issue tracker instead of both the tracker and the physical board, but went back to using the board as well as this worked better.

Use of the board as a coordination and communication tool is mentioned several times in several interviews, and the PO and the project leader are particularly fond of it:

> *"Yeah, we use a physical board. And that is clearly to be recommended. Anyone who doesn't have one, they shouldn't be allowed to call themselves a Scrum project [laughs]."* [A1-314]

> *"I'm particularly fond of them [the physical boards], because it's so easy for me to get an overview. I see how many cards there are in every column, and there's a magnet on each with the initials of the person who's working on it. So it gives a very good overview for those who understand a board like that, and who know the developers, so that they can ask questions whenever needed."* [A5-1550]

All stories have a task attached named "QA with Product Owner". This may be compared to an (informal) acceptance test.

> *"We've made.. that is, one quite big change [to Scrum]. We've said that "okay, we have to have a round of QA, for every story, before Review". Previously, the review was the first time I ever saw the finished result. But now... the Review has been toned down somewhat, since we've had QA on all the stories during the Sprint."* [A5-841]

Whenever a story is finished, a developer contacts the PO, and the QA is held:

> *"That makes it a lot easier for me, because then I don't have to do everything at the end of the week. So we get stories ready for QA the first week, and sometimes as early as Wednesday in the first Sprint week."* [A5-855]

At the Sprint Review, everyone from the project team is present. External stakeholders may be present when relevant stories have been developed. It is sometimes difficult to make them attend, but this has improved lately:

> *"That's motivating for the team as well, that people who throw in requirements actually get to see that we've developed the functionality, because a lot of times people just throw in the requirements and then they don't care about the end result, and that's pretty demotivating. But it's like that in a large organization, and it's understandable, although it's sad. And lately we've actually succeeded a lot more often with that [getting the stakeholders to attend the Sprint Review]."* [A1-527]

Before the developed stories are deployed to production, the functionality is tested manually by several testers who are brought in a few days of the week. A-3 (the developer) mentions that one of the biggest problems throughout the project has been getting qualified personnel for this work. At the start of the project, deployment was scheduled every third month, but the time between deployments has decreased until it is now two weeks. Deployment happens after the Sprint Review.

All the interviewees mention the need for extensive communication throughout the development process, both between developers and between the developers and the "business side", which includes the interaction designer(s). Close co-location is stressed as an important factor:

> *"If you want to succeed with Agile development, close co-location is alpha and omega. You have to sit in the same zone. People should have their desks right next to each other. A distance of as little as 10 meters makes it more difficult to communicate."* [A5-278]

Several interviewees mention that the team might change their process to Kanban some time in the foreseeable future, partly because the main development project is in its last stages. A-5 states that

> *"I've always thought that Scrum works best in large projects, and that Kanban maybe works better in operations, where there are usually smaller tasks that aren't a part of a larger context."* [A5-1236]

5.2.3.2 UX work, activities and related themes

5.2.3.2.1 User needs, research and design

Most of the original goals of the project were focused on providing a better customer experience on the web. The PO (A-5) was one of the initiators, and he has extensive knowledge about customer needs and wants due to his background in Customer Service. While there were several rounds of up-front requirements gathering in the original, large project started in 2004, most of the requirements for the integrated self-service system developed the last few years were based on having to replicate functionality present in the previously used systems.

For several years, the whole backlog was displayed in the form of post-its ("super-stickies") on a wall, the color and placement of the post-its reflecting issue/request types and prioritization. The backlog items were created mainly by relevant stakeholders in the organization, working closely with the PO and the interaction designer. The stories were then "translated" by the interaction designers into mockups, flow diagrams and other information needed by the developers.

Recently, prototyping tools have been used to create "clickable", interactive prototypes to show the flow between the screens. However, a communication tool that has proven particularly useful is having the whole workflow of a particular function as an "exploded" diagram on the wall, complete with mockups of the screens for each scenario. That way, the real complexity of the flow is shown, instead of a single "happy day" scenario. As some of the flows include many steps and several kinds of input from the user, the complexity of even a single flow can be extensive. Discussions between designers, developers, stakeholders and the PO frequently take place in front of the diagram, as it supports a common understanding of the functionality that is to be developed.

The project has been going on for several years, and most of the features that are currently being developed are in some ways variations on a theme. According to A-2, some written design guidelines exist, and some guidelines are just "common knowledge", documented only in prototypes and on the web pages. Designing the UI of new features largely involves re-using existing design elements and following existing patterns, although data structures and similar in the back-end systems may provide significant hindrances.

While the project has encompassed large parts of the organization's web site, another project is responsible for pages that do not require advanced back-end operations. Both A-2 and A-4 repeatedly mention that there has not been enough coordination and communication between the UX people of the two projects (and other people working with projects that affect the overall user experience on the web site). A-2's new role as UX lead for web is one of the measures taken to remedy this situation, and he has instated weekly meetings for all UX personnel working on web-related projects. At the time of writing, only four meetings have been held, but both seem positive that the meetings along with a more conscious focus on the coordination issues will improve matters.

5.2.3.2.2 User testing/evaluation

The company hires an external company to do user testing, and there have been five or six large, traditional user tests over the years. In the most recent test, fifteen people from the organization observed the testing (or parts of it), including developers, internal stakeholders, interaction designers and the PO.

More testing was performed in the initial phases of the project. The overall concept and the main navigational structure were validated in early usability tests. Later in the project, design decisions were based on the information gained from these combined with what was learned from the actual usage of the services on the site.

> *"We did a lot more of it [user testing] in the beginning, and then there's been less and less... and we should have done more of it."* [A4-818]

Testing has been performed using working prototypes and actual working code, but not on paper or other low-fi prototypes. When asked about whether informal user testing ("guerilla testing") is ever done, A-2 responds

> *"Like "quick and dirty"? [...] I guess we've all done that far too little, maybe? When that's said.. I'd have wanted more. Now I'll be making the usual excuse, "but there isn't time" [laughs], but there's really not."* [A2-569]

A-4 states that he would have preferred A/B-testing combined with the use of statistics from actual use and guerrilla testing to the traditional testing in a laboratory, but that they haven't been able to do that, for several reasons. One is the lack of UX resources, but it is also related to the way prioritization works on Agile projects, as it is not always easy to prioritize features (like infrastructure for A/B testing) where the benefit is mostly reaped in the long term, versus features that have significant short-term benefits.

> *"[...] and there's a lot of focus on "we need to deliver functionality". Must deliver quickly. And then it's like "no, we need to deliver.. that function, that has higher priority now". And that's sometimes.. a challenge, the way prioritization is done in Agile projects, because right now, that functionality is important. In the long term, we'd have had a lot better product if we'd been able to prioritize the A/B testing, in the beginning."* [A4-846]

> *"And there's also.. limited resources, one interaction designer for two teams. If I'm supposed to use.. two weeks to do a full traditional user test, I won't be able to finish the things they need for the Sprint."* [A4-908]

A-5 tells that they have recently developed a system to make it easier to do informal user testing, as well as demos:

> *"No, we haven't [done any informal testing]. Maybe we should have. [...] We've prepared more for it now, the application can be zipped down and put on a memory stick, so we can start it and run it using mock data anywhere. [...] [This was developed] so that we can run user tests. And it'll also be a lot easier to run guerrilla user tests."* [A5-1427]

5.2.3.2.3 Tasks

UX-related tasks have not been tracked on the Scrum board until now, although that might change, as the interaction designers are being increasingly more integrated with the team. So far, the user stories in the backlog have in many ways served as the task list of the interaction designer, as there is usually a need to prepare mockups or other kinds of UX-relevant information before pre-plan 1 or pre-plan 2. When the developers break down stories into tasks, some of the tasks might be assigned to the interaction designer(s), like "create mockup" or "clarify...". These are tracked in the same part of the issue tracker as the other tasks, attached to the relevant story.

When asked whether it's easy to keep track of UX tasks, A-4 responds

"Well, it's been.. a bit on and off. Sometimes you lose a bit track of the tasks, and sometimes.. I guess it depends a bit on how well defined the user stories are, and whether they're small enough, because I often work with many stories in parallel." [A4-613]

A-2 has mainly been working on another, closely related project. This project has been using Scrum until recently, when they switched to Kanban. In this project, UX tasks are tracked on the card wall along with all other kinds of tasks, and there is a separate up-front design phase that every story passes through in the workflow. A-2 has mixed experiences with task tracking from when they used Scrum:

"In my experience, in Scrum... suddenly a card was gone. I mean, everything was done so fast that sometimes slight adjustments were made that I thought were... very important, for the user experience, but maybe someone else would figure that it wasn't all that important, and just make changes that maybe looked really small from a developer perspective, but were pretty big in terms of user experience." [A2-921]

5.2.3.2.4 Role

The (various number of) interaction designer(s) used to work mainly with the PO and other internal stakeholders, and were not seen as team members in the Scrum sense. A couple of weeks before the interviews were conducted, the interaction designers were moved to sit among the developers, and they are now seen as parts of the team:

"The reason why we moved them into the team is that the developers have seen that it's more sensible if the interaction designers base their work more on what is technically feasible. [...] I think it's working very well, although they haven't been part of the team for more than two weeks. [...] Now *I'd say that they are a part of the Scrum team. We used to view their work as a pre-delivery to the Scrum team."* [A5-1062]

The interaction designers frequently work with people from the business side. According to A-4, one of the challenges is that products are being developed in the rest of the organization using traditional processes. When the team is tasked with creating the web pages and systems needed to sell the products, the products may be designed in a way that makes them hard to communicate or sell directly on the web. A-3 puts it simply:

"The interaction designers around here work a lot with the business side. That is, they work with the customers, both in order to help them understand what they can do, and the limitations.. the way it's often worked, is that all the limitations I've told you about, [...] the interaction designers end up explaining those to the business side. [...] The people on the business side don't understand those limitations, not because they're stupid, but because usually they haven't worked with this kind of stuff before." [A3-901]

However, the interaction designers don't need to translate between the PO and the team:

> *"The PO is very much able to talk to the developers himself, and he has a good technical understanding, so that.. there's not the same problem that there usually is in other projects, where you.. kind of have to be the hub in the middle, translating back and forth. If I'm not here, he's perfectly able to.. do that himself. And the developers.. if they have questions, they go to.. sometimes they come to me, sometimes they go directly to the PO. And I think that's a good thing.. if I get run over by a street cart, things won't stop completely."* [A4-586]

A-4 sometimes finds it challenging to keep a holistic perspective when working within an Agile process, although it's possible:

> *"That's the interaction designer's responsibility, to keep the complete product in mind, what is this going to be when we're done in half a year.. and make sure one doesn't slip, and start focusing only on the details.. [...] make sure that [the user interface] is consistent, that it keeps the overall focus that was envisioned when we started out. [...] It's possible, but it's not.. it's complicated. And you have to keep focusing on it, the process doesn't help you with that. [...] It's challenging sometimes, sometimes you forget, and then you have to.. you sort of sink down, and then you need to get back to the surface. And it might very well be that that's sort of okay."* [A4-636]

Communication between interaction designers and developers has not always been free of challenges. When asked how the designers and developers cooperate when mockups and workflows are created, A-1 responds

> *"Yeah, well, I guess that's been a weakness, there's not been enough of that, and that's why they need to sit with us. So that.. the developers who will implement it.. to see if it's feasible or not."* [A1-210]

A-4 describes that the process used to be more like a series of "mini-waterfalls". He used to help internal stakeholders write the user stories, and then "translated" the stories into mockups, flow diagrams and other information that the developers would need. The stories would then be discussed at the pre-plan meetings, and adjustments made according to prioritizations and the time and resources available. If it was not clear whether a proposed functionality/design could be implemented within the upcoming Sprint, the story would be moved back to analysis and exchanged for another user story with lower risk.

> *"[...]and then they would work on the story for two or three weeks or whatever the Sprint period was, with me supporting them. But.. often I wasn't used, maybe because I waited for them to contact me, and it might very well be that sometimes they waited for me to show up and work with them on the task."* [A4-164]

A-3 mentions that it takes some time for new interaction designers to familiarize themselves with the system and its limitations:

> *"[...] and when new interaction designers are brought in, they haven't.. they spend a lot of time getting to understand the limitations that are already there in the system. And that's the biggest hindrance. "Why on earth can't we just do it like this". Well, because dot dot dot and dot dot dot, we can do it, but then we'll have to postpone release by nine months."* [A3-727]

When asked whether they understand the limits when they are explained to them, he responds

> *"You know what, they understand them very well. The interaction designers here, they pull their hair and get less hair for every day that goes by [laughs]. That is, when you see.. when they get that harried look, you know that they've got it. [...] I don't envy the interaction designers the work they have here, because it's so complex. [...] And particularly because.. to handle that complexity, sometimes you might design around it, in some way, and then there's a lot of compromises, or then you need to take a fight, and that's.. you can get very tired from having to figure things out all the time."* [A3-742]

5.3 CASE B

5.3.1 HOW THE STUDY WAS CONDUCTED

The interviewee (B-1) was contacted through P-1, and direct contact was established by e-mail some time later. The interview was conducted at the premises of the consulting firm in which B-1 is currently employed, and lasted about 1 hour and 15 minutes. The interview was recorded, transcribed verbatim and coded according to the scheme presented in section 4.4. Some information was retrieved from the web pages of the organization and the consulting firm.

5.3.2 CONTEXT

5.3.2.1 The organization

The organization is a large department in the Norwegian public sector, with several hundred employees.

5.3.2.2 The project/product

The project was started in 2008, and is at the time of writing in its last phase. 175 people have been working on the project, of which 75 are employees of the organization and 100 are hired consultants. Developers from two consulting firms and the organization itself are organized in eleven Scrum teams, making the project one of the largest Scrum projects in Norway. It is expected that the project will finish on time and on budget.

The goal of the project has been to develop a new case management system, which replaces several older systems. The need arose when a legislative amendment made the previous systems obsolete, and it proved less expensive to develop a new system than to change the older ones. In the previous workflow, one of the problems was that a lot of the relevant information resided only on paper or in case managers' personal files, making it difficult to transfer a case from one case manager to another. Another problem has been that extensive training was needed in order to be able to operate all of the systems needed in the workflow. Both problems are sought resolved in the new system.

5.3.2.3 The interviewee

The interviewee is in his thirties, has been working with web projects since 1998 and as a UX professional since about 2000. He has been a part of the project for over two and a half years, and entered the project a few months after it was started. He has about six years of experience working with Agile projects.

5.3.3 RELEVANT FINDINGS

5.3.3.1 The development process

The development process is based on standard Scrum, although certain modifications have been made. Due to the size of the project and the large number of teams, additional roles and coordination mechanisms have been added to the ones found in Scrum. The 11 Scrum teams are divided in three groups, one group for each of the two contractors (consulting firms) and one group for the developers from the internal IT section. For every group, there is a Product Owner team consisting of the PO, a business architect, a functional architect and a technical architect. Every team has a Scrum Master.

In the first day of the three-week Sprint, a two-part Planning meeting is held. All the groups work in parallel, but the Planning meetings are separate for each group. In the first part, the PO presents the backlog items with the highest priority to the teams, and the teams forecast how many items they will be able to complete during the Sprint. In the second part, the teams discuss how the items will be developed.

The items in the backlog are presented as user stories to the team. The team breaks down the stories into tasks. During development, the tasks are tracked on a Scrum board (physical) and using an issue tracker (Jira). Each group of teams is co-located. Every day a Daily stand-up meeting is held, in which attendance is mandatory for the team members. In the last day of the Sprint, a Review meeting is held in each group, followed by a Retrospective. Developed functionality is released to production every six months.

The interviewee repeats several times during the interview that he is not particularly fond of Scrum, as he experiences it as "rigid", and because it doesn't take overall design and feedback into account:

> *"There are two things that are completely left out in Scrum, and that's what you're actually going to build, they don't say anything about that, they say build something and then iterate, the famous circle that everyone draws. And the problem about that circle is that it doesn't exist in the real world, because time is linear. When you start doing something, a piece of time goes by, and when you develop something, they say that now you'll get feedback, but it doesn't say* anything *about* how *that feedback is gathered, or how it gets back in the loop. [...] It's continuous integration that enables you to develop fast and release often, not Scrum. Even if all agile methods have an element of continuous integration in them, it's... people see Scrum as the standups and.. burndown charts and.. retrospective and all that fuss."* [B1-599]

5.3.3.2 UX work, activities and related themes

5.3.3.2.1 User needs, research and design

At the beginning of the project, a short design phase was conducted (termed "a sort of phase 0" by the interviewee). Personas were created, and workshops were held to create design goals for the solutions. In the workshops, stakeholders were asked to prioritize overall UI goals (flexibility, efficiency, error prevention and simplicity) according to preference. The UI goals differ for different parts of the system. The interviewee stresses the importance of having made a conscious choice of overall UI goals to guide the development, in order to preserve consistency throughout the system.

A release is developed in a time frame of six months. Before work starts on a release, the PO teams spend an intensive period of about 1-2 weeks creating an overall description of the functionality that is to be part of the release. The functional architect is central in this work, although the interaction designers are involved when needed. The functional architect works with the rest of the PO team and relevant stakeholders, and breaks down the backlog items into user stories. The functional architect also writes acceptance criteria.

Several highly knowledgeable case managers (expert users) are closely co-located with the development teams, assisting continuously with design and development work. While this has for the most part worked very well, it has also created some problems for the interaction designers. In the previous systems, the workflow has not been guided or restricted in any way, and experienced users have created their own personalized ways of using them. When users were chosen to participate in the project, the natural choice was the expert users who knew the existing systems very well. As the new system will support a guided workflow, it can be thought of as too restrictive:

> *"[...] so the newer case managers, they want a more guided workflow, more help along the way. And that limits the flexibility. So.. you get a resistance against introducing that, you know, among other things because those users who are chosen to participate in the project are so-called "old hands". They know too much IT, and they know too much about the domain."* [B1-197]

The interaction designers prepare wireframes and UI mockups using PowerPoint. No graphical designer has been involved, as the interaction designers are capable of doing the needed design work themselves, like designing icons. Design work happens both up front in the specification phase before development work starts on a release, and continuously in parallel with development. When asked how far ahead the design work happens in relation to development, the interviewee responds

> *"Anything from us being far behind, to a year or two in advance. [...] Suddenly someone from [the other consulting firm] comes to me and asks about some functionality we designed last year. That might happen. Or a developer comes and says that in this sprint we're doing that, so I need a button, which type of button should I use [...]"* [B1-792]

The interaction designers are seated at the same table as the developers. Sometimes, UIs have been designed by a designer and developer sitting together – "pair design":

> *"So the developers called it "beautiful GUI", and that included a kind of pair design, depending on how you see it, with us. That we just sit there beside them and hold their hand when they create the buttons and stuff."* [B1-651]

The interviewee repeatedly returns to a central problem with design work in the project; that there has been little room for choosing UI solutions that require more than a minimum of development work, on account of the strong focus on delivering functionality on time and on budget. A lot of the interaction designers' work has consisted in making sure that the "least bad" UI solutions were chosen.

> *"We tried as hard as we could, you know, but we just had to grin and bear it... you know, we'd say "you should do this, and the optimal solution is this one". And then the architect would say "no, we can't do that, we need to scope it down, just use a simple checkbox to reveal the information". And we'd answer something like "but that's a bad solution, only experts will understand that". "Yeah, but...", you know. And of course, it could have been solved by pouring in millions more into the project, in theory. In practice, I don't think it would work, you'd get all that communication overhead and so on. But.. they have a budget, you know, the Norwegian state is paying..."* [B1-454]

5.3.3.2.2 User testing/evaluation

Three usability tests on representative end users have been conducted, in addition to continuous design evaluations performed by the expert users on the project. All of the tests were conducted using actual working code, although the functionality had not yet been released to production.

The interaction designers encountered several problems with doing user testing. The first was to gain access to representative users, as the project had only budgeted for the use of the expert users' time. For the first test, the designers originally asked for 10-12 subjects with varying degrees of IT and domain expertise, and planned for the test to be conducted at the premises of the project. After some time and discussions, they were given permission to visit case managers in their offices, and conducted on-site testing with six subjects. The test provided highly useful information for the project, and the designers had more resources available for the second and third test:

> *"At the third test we had.. we got two adjacent meeting rooms, we rigged a video link between them and had lots of observers from the project. And we had a good range of subjects, and we got them to come to us, so we could spend two hours and not one."* [B1-253]

The interviewee also mentions that the developers that observed user testing for the first time gained a very favorable attitude to the practice, becoming testing "ambassadors" internally on the project.

Another main problem with doing user testing was that there was no room for actually making use of the results, to change functionality that had already been developed. This put an end to the testing, as the designers saw no point in gathering data that would not be used. All of the tests were conducted during the first year of the project.

> *"And then there's the problem with us doing a user test, you get a lot of feedback, and then we'll want to change things that didn't work. And after the third test we had a lot of*

> *things we wanted to change [...] we thought Scrum would be fantastic, because you'd get the chance to change things, you know. And.. when we asked to change things, they said no. "We haven't got the time or the money, we haven't really got the time or the money to have you on the project either" [...]"* [B1-430]

> *"We didn't see the point of spending time doing [user testing], when we knew that improvements wouldn't be made anyway. Or they wouldn't actually redo things, even if we thought they should. But we used the results from the old tests, to the fullest extent, to convince people of this and that, and we used them to guide our own... gut feeling of what was important."* [B1-905]

When the interviewee was asked whether it was difficult to find the time to do user testing, he responded *"No, it's more difficult when you're only one [interaction designer]. Then it's very hard."* [B1-903]

5.3.3.2.3 Tasks

The developers track their tasks on a physical Scrum board as well as in the issue tracker Jira. The UX tasks are not tracked on the board, although the developers have a type of task called "beautiful GUI" that signifies the need to get designer input on the UI.

At one point, the designers tried adding tasks to Jira. However, this was regarded as noise, and the designers were given a separate part of Jira to put their tasks in:

> *"Yeah, it was regarded as noise, when we started creating Jira-tasks, and they started to appear in prioritization meetings and stuff. After a while we got our own project code where we could create issues, and if a developer would have a* lot *of time on his hands, maybe something would be done about them."* [B1-885]

5.3.3.2.4 Role

The number of interaction designers has varied throughout the project. In the group the interviewee belongs to, the number has varied between one and three. Currently only the interviewee is left, and he will be off the project a few months before the final release.

> *"When we were three, we had one who sat with the overheads, the project leader and the functional architect, and did the workshops with the customer and stuff. My colleague and I sat on separate teams, and rotated between the three, as we weren't three persons.. we should have been. And in addition.. everything doesn't always go according to plan, and [the other consulting firm] ended up doing some GUI as well. So we had to serve them a bit too."* [B1-634]

While there were only three people focusing on UI-related issues, no formal coordination mechanisms were needed – “we just talked over lunch”. But for a time, many features being developed required a lot of UI work, and several teams across groups worked on UI-related stories. The designers identified a developer on each team that was interested in UI-related issues and likely to do UI development, called “stylists”, and designers and stylists from all the relevant teams across all groups would meet two or three times a week.

> *“And then we would meet, have a quick stand-up two or three times a week. And that worked very well for a while, as there was a lot of GUI work going on in many places. So a guy from [the other consulting firm] would show up, and one from [the organization], when they were working on a large task having a lot of GUI. And they would tell us what they'd done, and we would say “okay, if* you're *doing that, then* you *can't do it the other way, you know, it has to be consistent.. [...] So we coordinated that.”* [B1-811]

Sitting close to the developers is seen as advantageous:

> *“There's something to be said for actually sitting close to the developers. We quickly saw that when we weren't sitting at the long table, literally, where the team was seated, we weren't asked as many questions. By sitting there, it was like “[B-1], how was that supposed to work, should the user be able to drag that panel”, and stuff like that. And you get a lot better feeling for what's actually possible to do, and so on.”* [B1-686]

Although they were closely co-located with the developers, the interaction designers were not part of the actual Scrum teams. This was sometimes problematic:

> *“We'd get questions... you know, sometimes, when they thought it was relevant, we would participate in the daily stand-up. And sometimes they would start nagging about a card on the board, and we'd get questions like “are you part of the team or not, can't you just make up your mind”. So I guess it might have been a bit hard for them to cope with it.”* [B1-695]

However, the developers are pleased to have interaction designers as part of the project team:

> *“Oh yes, they do [respect our competence], and they think it's really nice that we're there, because they can.. they can relax a bit about it [GUI work]. I think that's a bit.. wrong. Because we've often peered over their shoulders and said "hello, aligned grid.. on that side it should be like that.. you have table rows of different heights...* think"*, you know. But of course.. they think about other things. So I think they feel it's very nice that we're there.* Because *we're there, we get some... sloppiness, because they expect us to catch things. And when we catch things.. they don't necessarily learn a lot from it, but they might relax more, as they know we'll catch things.”* [B1-841]

> *"And they see that it's very practical to have the mockups, to get everyone on the same page, like "oh, so* that's *what they mean". And the mockups might be wrong, but it's like.. that's valuable in itself, to know that "this is not what we wanted", before it's actually developed."* [B1-855]

Overall, the interviewee feels that the designers had to press hard to get their opinions heard:

> *"[...] we had to do, like, guerrilla work, pushing things we thought were very important. So we quite simply had to try to.. almost bribe the developers to get it done. Or we would continuously nag for something, or by pointing out the stupidity of certain solutions manage to persuade people that.. that solution should be chosen, not necessarily because it's the most user friendly one, but because there are.. many advantages to it."* [B1-894]

5.4 Case C

5.4.1 How the study was conducted

The interviewee was contacted directly by e-mail and asked for a preliminary interview. This interview was held some time later, lasted about an hour, and was mostly unstructured. No recording equipment was used, although extensive notes were taken. The interviewee agreed to do another, more structured interview later on.

Before the second interview was held, another student writing about Kanban (the same as in case D) contacted the interviewee. It was agreed that the interview should be conducted by both students together in order to save the interviewee's time, as large parts of the question sets would overlap. The interview guide was created collaboratively by both students. The interview lasted about two hours, was recorded in its entirety and later transcribed verbatim. Both interviews were conducted in the premises of the agency in which the interviewee is employed. The interviewee was asked for pictures of the Kanban boards, which she kindly provided some time later by e-mail. Additional material used was publicly available information on the project and organization, retrieved from the customer organization's web site.

5.4.2 *Context*

5.4.2.1 The organization

The organization is a large department in the Norwegian public sector, with several thousand employees. The ICT section in the department hires external expertise from consulting firms to do large development projects.

5.4.2.2 The project(s)/product(s)

The development effort is part of a large project with the overall goal of making it easier for both internal and external stakeholders to save, retrieve and share information in a simple, secure and effective way. The project was started in 2006, and several large systems are being developed/re-developed and integrated as part of it, including the organization's intranet.

The system being developed is a case management and document handling system. When released, its user interface will be embedded in the organization's intranet. As with many systems developed for use in the public sector, numerous requirements are imposed on the system as a result of laws and regulations. The interviewee repeatedly states during the interview that the system is "enormously complex", partly resulting from having to comply with requirements for electronic recordkeeping in public administration.

5.4.2.3 The interviewee

The interviewee (C-1) is working as an Interaction Designer in a business and technology consulting agency of about 300 employees, and is currently spending 60% of her time on the project, working as part of a development team hired from the agency by the organization. She has the equivalent of a master's degree in Product Development, and has a total of seven years of work experience. In the two first years she worked in traditional projects (stage-gate/"waterfall" process), the next four and a half years in Scrum projects, and she started using Kanban half a year ago (at the time of writing). A few years ago she became a certified Scrum Master, having attended a course held by Ken Schwaber (one of the Scrum creators). She is very interested in the subject of UX/Agile integration, and has recently promoted Kanban at various conferences.

5.4.3 Relevant findings

5.4.3.1 The development process

The organization hosting the project has hired the entire technical/UX team in addition to a project leader from the agency. Until very recently the developers were organized in two teams, each having its own Product Owner. Currently there is one team of six developers, and a smaller team of three developers mainly concentrating on operations/maintenance issues. There are two UX resources on the project, C-1 (60%) and a junior interaction designer (100%). The hired team works closely with a PO and five domain experts (some part-time) from the hosting organization, and all are co-located at the same premises. The main development team, UX resources and the PO are seated in the same room.

This part of the project switched from Scrum to Kanban half a year ago, the interviewee being the main initiator of the switch. They have kept "the things that worked" from their Scrum process: the Daily stand-up meeting, the Scrum Master role, and the Review and Retrospective meetings held at two weeks' interval (although the "Sprint" concept itself has been dropped). The Daily stand-up takes place in front of the card wall. The PO now joins in every day, and the traditional three questions have been replaced by discussions centered on the user stories currently on the board.

The columns used on the card wall are "Specification" (sub-columns "In progress" and "Done"), "Design Meeting", "Development", "Test", "QA" and "Accepted". The Work In Progress (WIP) limits have changed from time to time, but at the time of writing the limit was six for the specification column, five for Development, three for Test and three for QA. The Development WIP limit has varied according to the number of developers working on the project. Post-its and custom printed forms are used as cards. A user story spends on average about four weeks on the wall. The physical wall is replicated by a digital card wall system (Jira issue tracker with Greenhopper plug-in). The Scrum Master is the main person responsible for keeping the physical and digital walls in sync.

When a task enters the specification column, it is usually in the form of (large), roughly described user stories or epics. A workshop is then initiated by C-1 and the PO, in which the PO, the interaction designers (usually both), 1-2 domain experts, the Scrum Master (often) and 1-2 developers who are likely to be the one(s) working on the story later, participate. The developers are often chosen based on their individual skills:

> *"Many say that in a team everyone should be able to do any kind of work, but it's not like that in a development team. People have their specialties, that's just how it is. Some have worked a lot on one part of the system, some have worked on another."* [C1-370]

On average, 2-3 workshops of this kind are held every week.

During the specification phase, the epics and stories in their "rough" form are divided into smaller stories of about the same size, refined and enough information is added to make the story ready for development. Usually several meetings are required:

> *"[...] we have several meetings like that before we can write the final user stories and create the user interface mockups, because several new things always pop up in every meeting. Like "we didn't think of that" or "oh, that's not possible" or things like that. [...] Things stay pretty long in the specification phase, since the [project] is highly complex."* [C1-441]

The interviewee often returns to the subject of user stories:

> *"We use a lot of time to write [the user stories], we've realized that good user stories are extremely important. Even if everybody participated in the specification phase, people remember what was said differently, so it's important to write it down in fairly numerous and clear acceptance criteria."* [C1-492]
>
> *"Because it's an extremely difficult art to write user stories [laughs], you can't just let anybody do it, you need a lot of experience, and you need a lot of experience as to what you can write as acceptance criteria, how you can write them, how the team interprets them, what are good and testable acceptance criteria, what is design, what is functionality, what should be included in the user story and what shouldn't.. and how you phrase it."* [C1-516]

According to her, stories should be as small as possible, while still adding value to the user. The title and the acceptance criteria are very important, while the actual content (the "as a ... I want ... so that ..." part) is "never read by anyone anyway".

The final user stories are usually written by C-1 and the other interaction designer. The PO would like the domain experts to write user stories as well, but so far it hasn't been possible for them to do it alone:

> *"The PO wants them to write user stories, but that won't work because the quality won't be good enough.. so at least in a transitional period we'll have to do it together, we [designers] know how to write them. Also, you have to have been a part of the process, design- and specification process, all those meetings, in order to know which acceptance criteria to write in the user stories."* [C1-529]

Sometimes the developers write their own user stories, but according to C-1 this isn't always a success:

> *"Yeah, they write their own user stories sometimes... [sighs] There are both advantages and disadvantages to that. Sometimes they write stories like "see to it that this technical process works like that", which aren't really testable from the user interface. I*

> *understand that they need to [write stories] sometimes, when the total amount of work on a story is too much if it's not divided in smaller stories.. but generally we try to write stories that include all layers."* [C1-732]

The main problem with having "technical" user stories is that they are difficult for the PO and interaction designers to accept in the QA phase, as testing might depend on a part of the user interface that is yet to be developed.

When a user story is ready for development, it goes in the "Design meeting" column. In this phase, the developers who intend to work on the story meet with the interaction designers before they start coding. The meeting ensures that everyone (still) agrees on what is to be done. Developers and designers also communicate and collaborate during the "Development" phase.

When the developer(s) finish working on a story, it is placed in the "Test" column. Another developer then performs a code review before the story moves on to the "QA" (Quality Assurance) column. The PO is then responsible for testing the story and accepting or rejecting it according to the acceptance criteria, although the interaction designers often help with the testing. It has been suggested that the domain experts should participate in this phase, but this is not yet the case.

> *"Quite often we'll write comments on the user story, and then it'll be moved back to the specification column if we discover bugs or mistakes. [...] Usually it is placed at the top of the column just because it's important to finish the stories [...] But it might actually go a few rounds, but that's what's iterative [laughs]. Sometimes one might think of something new, and when that happens it's important to create a new user story instead [...] so that stories won't loop forever on the wall."* [C1-649]

When stories are moved back to specification, a pink post-it with a note of what needs to be done is added to the card. This has the added benefit of providing a clear visual cue to which stories on the board are currently "in a loop".

Although the development process in itself is agile, deploying code to production servers is subject to a more formal process defined by the ICT section. Very little of the finished code has been deployed to date.

> *"You need to write a change request, that needs to be written two months in advance, and you have to write it completely right in order to get it approved, and it's generally.. cumbersome."* [C1-260]

> *"No, we haven't deployed very much. In my defense I'll have to say that.. the case management, at least as far as the end user is concerned.. there hasn't been much of a point to deploy anything up until now, as they can't actually manage any cases, but soon they'll be able to do certain things, at least."* [C1-1222]

The team spends considerably less time estimating effort now, compared to when they were using Scrum. The planning meeting is no longer held, and the current form of estimation is based on how many user stories a task/epic might consist of. The user stories are kept approximately the same size, and the team knows how many user stories they usually finish in a month's work.

5.4.3.1.1 Changing from Scrum to Kanban

When asked why the team decided to change their process from Scrum to Kanban, C-1 starts by describing how Scrum didn't work for the team:

> *"The team almost never delivered what we said we would deliver, and we spent a lot of time on planning. One day of sprint planning for a two week sprint, in addition to the sprint review. That's a pretty high percentage of the time, and the results weren't very good since the developers weren't part of the specification phase, so a lot of issues would pop up during planning that weren't... that is, technical issues like "this can't be solved like that", so we had to spend a lot of the sprint to figure out what was actually supposed to be done, or make changes, and so we weren't able to deliver what we were supposed to in the sprint."* [C1-849]

This created a vicious circle where the team was constantly behind schedule, lagging ever more behind for every sprint. Even though they were all co-located, there was not enough communication between the team, the designer and the PO during the sprints, which also led to problems:

> *"[...] there's no common goal, no communication, no common understanding of what is actually being developed.. so sometimes they [the developers] would make their own assumptions, and then make something that wasn't like it should have been at all [...] the PO was a lot less involved in the actual development. Issues weren't discovered until the review meeting, in which it would be like "oh, this went wrong" and things like that."* [C1-861]

After the process was changed, the team experienced considerable improvements:

> *"Communications have increased by... 400%, I'd say [laughs]. It's especially important that the PO participates in the daily stand-up, that everyone participates, that everyone knows why something's blocked, why something progresses slowly, what we can do to improve that. [...] And the specification phase is* completely *different.. that we actually get the time, have the time, and that people actually know what we're supposed to do.. and the user stories are a lot better now."* [C1-1021]

C-1 also mentions that the improved quality of the user stories partly results from getting another (junior) interaction designer full-time on the project, who spends a lot of her time writing and making changes to the user stories.

Later C-1 also mentions that the level of conflict has decreased, that the developers experience a stronger feeling of ownership of the stories, and that the perceived development speed has increased.

> *"I think the conflicts... it's not that there used to be a high level of conflict, but if there was, it's definitely not high now, between the developers and those creating the specs. Because they've participated in the spec phase now."* [C1-1170]
>
> *"Everybody experiences that we're developing much faster now, but I can't say whether we actually do. Before, there was a lot of focus on low speed... and now there isn't."* [C1-1181]

Overall, the interviewee is extremely pleased with the new process:

> *"I just feel that the more I work in the new process, oh my god, why did I toil with Scrum this long? [...] And especially for [the non-technical developers], to get the time to specify what you're supposed to do, Kanban works a lot better."* [C1-1266]

5.4.3.2 UX work, activities and related themes

5.4.3.2.1 User needs, research and design

In the initial phase of the project (2006-2007), numerous workshops were held to map the overall requirements of the system, but the work was not focused on user experience:

> *"I know they ran a lot of workshops on, like, "my dream work space", and things like that, I've seen quite a few powerpoints with.. what they want, but that's more related to case management. [...] But the big, overarching analysis of how people actually work, how should they be able to work, and how do they work now, that hasn't been done."* [C1-1400]

While the overall functionality was specified at the start of the project, additional and more detailed needs and requirements are gathered mainly through internal resources on the project. C-1 feels that there should be more communication with users outside the project:

> *"[...] we have some case managers within the project, so they can participate in workshops and specification meetings, or we do user tests, or interviews with case managers, but I have to admit that we haven't been very good at gathering information from outside the project. [...] it's always important to talk to the end users. I'll take the blame for that... we should do more, do thorough user interviews and such."* [C1-1104]

Most of the detailed design work happens in the specification phase. However, overall conceptual work is done before the tasks reach this phase, although there is currently less work of this kind now than before:

> *"Design work like that happens pretty much in the specification phase. But things like "flow in the main concept", that doesn't really belong to a specific user story.. and like, when we have to look at how search relates to the task lists, and things like that.. We don't do that in the specification phase, those are things that are done when we do analysis work, to get* to *the backlog, to get* to *the specification phase."* [C1-1322]

> *"I think maybe I spent more time doing overall conceptual work before, like maybe a year ago or something like that. Now we know the overall principles fairly well, the main concept is more or less there."* [C1-1333]

In addition to mockups and detailed acceptance criteria, flow charts and similar are also created to support understanding of the overall flow, and how the user stories are connected. These are attached to the user stories as supporting documentation in the issue tracker.

C-1 spends most of her time doing analysis and concept work, while the other interaction designer is "more of a doer", creating mockups and refining user stories. A graphic designer is brought in when needed, although current work mostly involves using elements that have already been designed.

The team rarely makes changes to features that have been developed previously, and this slows the process down to a certain extent:

> *"We iterate very little in the project, because there's a lot of pressure to deliver functionality, and things are slow because of dependencies and stuff like that.. so if you've gotten functionality deployed to production, the chance that you'll be able to improve it is very small, and that makes you do all you can to make it as good as possible before deployment. And that means things take even longer to reach deployment, and people end up desisting deployment."* [C1-826]

5.4.3.2.2 User testing/evaluation

User testing on actual working features in the production environment has so far not been possible, as most of the developed functionality is still to be deployed, but some testing of basic case management functionality has been done using the development environment. Paper prototype testing has also been used during UI design.

> *"We've user tested the big things. But of course we could have done more of it! But it's difficult to change functionality once it's been developed. [...] But paper prototype testing.. that works. That's very good, it can be done very fast and easily, and it only affects the design phase. That makes it a lot easier to make use of the suggestions."* [C1-1358]

For a time, a group of case managers were involved in the project, and C-1 found this highly useful. The group was removed from the project, but the team still has access to several case managers who are located in the same building as the project team, and these are sometimes used for input.

> *"We're lucky to have the users at the premises. [...] So we can.. sometimes we just go to their offices and ask "hi, are you a case manager?" "yes" "can we ask you some questions?" ... and then we'll stay there for an hour, they love talking. [...] But I think maybe the end users should be involved a lot more."* [C1-1374]
>
> *"The big user tests where you learn a lot are important, of course, but it's even more important to do the continuous.. where you can just ask.. in the design phase."* [C1-1388]

5.4.3.2.3 Tasks

The day before the interview was conducted, the two interaction designers and the PO had set up their own Kanban card wall.

> *"Now we've set up our own Kanban card wall, because we all had our own private lists, the PO, [the other interaction designer] and me, with "we need to do this and we need to do this", and in the end we figure that we needed to gather all the lists and set up a Kanban card wall. That's our own tasks, so we have our own work flow."* [C1-1237]
>
> *"But it's very important that the developer Kanban card wall is the main card wall, that we focus on. [...] We'll never put a task like "write user story" on our wall, that is a part of the specification phase on the main one. But it could be tasks like "talk to case manager to get information about that one".. like tasks we just need to do."* [C1-1250]

The three have defined their own task categories:

> *"[...] so we defined the task category "look into" [laughs], and "workshop", and.. "epic" [...] which is mostly for deciding.. that we need to do something about this functionality.. and "design", mockups and things like that. Those were the four main categories we identified. And one called "user stories" as well, for tasks like gathering information for the user stories."* [C1-1288]

5.4.3.2.4 Role

C-1 works very closely with the Product Owner and project management, and participates in most of the overall planning work. While the developers show increasingly more interest in the product as a whole now that they participate in the specification phase, she still feels that she has to be "the user's advocate", while project management and developers focus more on the importance of technical issues. But the combination of the two perspectives is seen as fruitful:

> *"The tension promotes creativity. You sort of get to use both perspectives, and in the end I think the result is better than any of the original ideas."* [C1-1502]

C-1 has been using the parallel track solution for a long time, during the Scrum period. When asked if or how this way of working has changed as a result of moving to Kanban, she replies that what *hasn't* changed is the way she needs to do all kinds of work all the time, switching constantly between overall conceptual work and nitty-gritty details, design and testing.

Although she has experienced some minor communication challenges on previous projects, she feels that UX people and developers generally work very well together.

> *"Yeah. So I don't think there's a "battle", or a conflict of interest. People understand that they have a common goal, it's just that they have different perspectives. And I think.. I never experienced that either, I think everyone is constantly talking about "oh, developers and interaction designers, they don't speak the same language, there's such a high level of conflict". I never experienced that, don't know what kind of people do. [laughs] Can't people just talk to each other?"* [C1-1517]

> *"But of course it's like that, you read like, blog posts, people post things all the time like.. or maybe mostly 2-3 years ago, and even further back, like "oh.. designers and developers can't communicate at all", and there are only misunderstandings, and there's just no end to it. I think waterfall is mostly to blame. But in Agile too, it seems like people haven't understood that "oh, maybe we actually need to communicate". [laughs]"* [C1-1530]

5.5 Case D

5.5.1 How the study was conducted

Access to the project was gained through another master's student studying Kanban, and all meetings and interviews were attended by both students. First, a preliminary meeting of about an hour was held with the project leader. The project leader then arranged for interviews with the resident interaction designer, an experienced developer, the Product Owner and herself. Both the preliminary meeting and all of the interviews were recorded, although only the interviews were transcribed verbatim and coded according to the scheme in section 4.4. All the interview guides were created by both students working together, ensuring that all aspects related to both studies were covered.

All the interviews took place at the premises of the project, and lasted about an hour each. One daily stand-up meeting was observed, and permission was granted to photograph the Kanban board. The project leader also provided additional material in the form of a presentation, and the interaction designer provided examples of user stories. Additional information was obtained from the organization's web site.

5.5.2 Context

5.5.2.1 The organization

The organization is a large enterprise in the Norwegian private sector, with several thousand employees.

5.5.2.2 The project/product

The project started in January 2010. A few developers and a project leader worked for about a year, using Scrum (according to some of the interviewees, "Scrum in a waterfall manner"). It was then discovered that the project was severely delayed, and an audit showed that the delay partly resulted from the waterfall-style Scrum, as the focus had been on time spent rather than value delivered. A new project leader was hired with the explicit goal of making the project more Agile than it had previously been. Kanban was introduced in March 2011, and this has been the main process model since (more than half a year at the time of writing). The second project phase will be finished by December 2011, although development and maintenance work will continue in 2012.

The product integrates several specialized systems in the field of insurance, creating an integrated, holistic workflow for case managers with a web-based user interface. Where there were previously several different area-dependent workflows, the product provides a single work space with standardized workflows. This makes it easier to rotate personnel, and ensures equal treatment of customers. There are about 180 registered users of the product, all internal employees of the organization.

5.5.2.3 The interviewees

D-1 is the Project Leader, hired from a consulting firm. She has extensive experience with several processes, both Agile and more traditional, and has a master's degree in Information Technology.

D-2 is the Product Owner on the team. He has been working in the organization since 2003, and has been part of a previous systems development project, working with requirements specification. He holds the equivalent of an MBA, and has about two years of experience working on Agile projects.

D-3 is a developer, hired from a consulting firm. He has been working on the project from the beginning. He has about two years of experience working within Agile processes, and has the equivalent of a master's degree in Computer Science.

D-4 is the UX specialist (interaction designer) on the team, hired from a consulting firm. At the time of writing, he has been part of the project for about half a year. He has worked with Agile processes for about 3-4 years, and is a certified Scrum Master. He has a varied educational background from several institutions (university/college), focused on creative and technical subjects.

5.5.3 RELEVANT FINDINGS

5.5.3.1 The development process

The project team consists of 9 project workers and 4 expert users. Among the project workers, there is one project leader (who also acts as the equivalent of a Scrum Master), a Product Owner, an interaction designer, a functional architect and five developers. The PO, the designer and the functional architect form the “functional team”. There are currently four back-end developers and one front-end developer. Every other week, a team of four expert users spend two or three days in the project offices, testing developed functionality and discussing designs, features and new requirements.

The development process is based on Kanban. A physical card wall in the offices provides a visual overview of the workflow from specification to done. Every day, a stand-up meeting is conducted in front of the card wall, and features currently in development are discussed. Both the project leader and the Product Owner attend this meeting. The team no longer uses the standard questions asked in the Daily stand-up meetings in Scrum, but rather focuses on sharing information about work currently in progress and removing impediments.

Every card on the wall represents a user story. The card contains seven types of information: a backlog ID, date, Sprint number, release number, estimate, description and task type. The backlog ID is used to locate additional information about the story in the project wiki (Confluence). The description is on the user story form ("as a ... I want ... so that ..."). Stories may be moved both forwards and backwards on the card wall. When a story does not pass one of the checkpoints (code review, sit-down, test), it may be moved back to the appropriate phase. Stories may also be reprioritized or removed from the card wall at any time, although this is not supposed to happen too often. The stories are estimated using story points, and sizes may vary. On average, a story spends about a month on the card wall from start to finish, while the coding phase is usually somewhere in the range of 3-4 work days. The specification phase takes up a significant portion of the total time.

The workflow on the card wall currently has 8 phases/columns. The first is Specification, in which stories are elaborated by the functional team. When specification work is finished, the stories move on to the "Ready for development" column and are sorted in order of importance/priority. Developers then pick stories from this column and move them into the "Development" column when work is started. When development is finished, the story is moved to "Ready for code review", from which another developer picks the story and reviews the code. When the developed code passes the code review, the story goes on to the "Ready for sit-down" column. Every morning after the stand-up meeting, the stories in this column are reviewed by the PO (or the interaction designer) and the developer. If they pass the review, they move on to "Ready for test". Code belonging to stories that are in this column is regularly deployed to the testing environment, and the stories are moved to the "In test" column. The four expert users test the stories, and move them to the last column, "OK in test", if everything is OK.

There are also a varying number of "swim lanes" on the board. One of them is currently dedicated to technical tasks (called "technical debt"). When developers decide that there is a need for technical work that is not necessarily related to a specific user story, a card is put in this swim lane, and the cards are discussed and prioritized regularly.

Currently, no specific guidelines or criteria exist for when a card is to be moved from one column to another. However, plans exist to introduce an "acceptance test" between the

Specification and Ready for development columns, in which a developer checks that the specification is clear and complete enough for the story to be ready for development. Also, the criteria for completing a code review will be extended, so that the review not only encompasses a review of the code, but a functional test as well to see that the feature actually works.

As the product is in production and in regular use by the case managers, bug reports and maintenance issues crop up continuously. In order to shield the development team from the "noise" of such issues, while still being able to respond quickly, a developer in the role of "guardian" handles all such issues. The role of guardian is rotated among the developers, each having the role for a week before it is passed on. Both the project leader and the developer mention that this works well.

Code is released to production every month, on average. Bug fixes and updates may be deployed even more frequently. The Product Owner, the project leader and the developer all mention that the frequent releases have improved the team's internal standing in the organization, as stakeholders have repeatedly seen that their requirements are developed quickly and to their satisfaction.

All the interviewees repeatedly mention that the card wall is a very valuable communication and coordination tool. In the words of the project leader and the interaction designer:

> *"The card wall has been incredibly valuable, I think. We've all become united by it, and it pulls us through our daily work. It's very visible, and now I think it's very easily understandable. I think everyone, no matter who you talk to in there [the project team], will be able to explain it, how it works and say something about that. So it's definitely our most important tool."* [D1-121]
>
> *"I think the most valuable thing we've had in this project is.. well, this is my opinion, but.. to actually visualize [the workflow] on a card wall is actually very simple, you don't need to make it any more complex than this, and it's very good for the developers and the rest of the team, and the project, to be able to see that the work flows."* [D4-476]

However, several interviewees mention that it has not been entirely challenge-free to reach the point where everyone takes responsibility for moving and updating his or her cards whenever work is started or finished, although the system has worked well for the last few months.

It appears that all of the interviewees are pleased with the current development process. The Product Owner and the developer both mention that the new process is more flexible and has increased communication between all team members and stakeholders, both internal and external. The project leader finds that the overall number of errors has decreased, and that an increasing number of errors are detected very early in the process, making them easier and less time consuming to correct.

5.5.3.2 UX work, activities and related themes

5.5.3.2.1 User needs, research and design

No UX professionals were involved in the project before the interaction designer was brought in about half a year ago. In the previous process, requirements were gathered and specified by internal stakeholders. The requirements were not written on a standardized form, and sometimes mockups of the UI would be drawn using PowerPoint. The developers created the UIs based on the mockups when such were available, or created them from scratch. According to the developer (D3), none of the developers on the team were especially interested in or focused on UI work, and it was clear that improvements could be made in this area. The developers did some user observation in order to understand work patterns, albeit a very limited amount.

The interaction designer (D-4) was brought in at about the same time as the new project leader (D-1), a little more than half a year ago at the time of writing. Initially, he observed users in situ, did in-depth interviews and held workshops with the expert users on the team, all in order to gain an understanding of the system and the work that needed to be done. The project is currently in a "UX-oriented phase", and UX-related work has been given priority for the next three releases. The interaction designer feels that he has been *given* enough time to do end user observation, although the amount of work in other areas necessarily limits the time available.

D-4 was partly brought in as a "coach" for writing user stories, as the PO and other members of the functional team had no prior experience in this area. Currently, D-4 usually writes most of the user stories, and he has set up a system using Confluence (a wiki system) to keep track of the stories and their status. When a story is in the "Specification" column on the card wall, D-4 either works with the functional team or by himself to provide all the necessary information to make the story ready for development, including mockups.

The actual UI design process varies somewhat according to the size of the task, but usually starts with designs on paper, continues in Photoshop, and eventually a high-fidelity prototype is often made using Adobe Flash Catalyst when the task is sufficiently large.

> *"I always use paper, I always start my work on paper. Yeah, before I do anything in either Photoshop or Catalyst.. I start on paper. [...] And it varies, depending on how large the task actually is, because you don't need to detail the entire flow if it's about a tiny*

portion of a page, where a field or drop-down menu is to be added, or something else. You don't need Catalyst for that. But here.. for the larger tasks, those that entail a completely new way of searching or a different architecture than before, or a suggestion for new architecture, [Catalyst] is used." [D4-266]

The developer is very pleased with having the user stories on a standardized form, and extremely pleased with the use of high-fidelity prototypes:

> *"[...] before, we weren't as good at specifying the tasks, and the worse the tasks are specified, the stronger the confusion when you're talking to a [member of the functional team] about something, because then both have to figure out what we're actually discussing. But now we have user stories, which are very specific, and so it's a lot easier to ask questions, and it's easier for the other person to understand what you mean. So that has improved a lot."* [D3-351]
>
> *"It's super, it's fantastic, because.. just getting the prototypes that quickly, it makes the job incredibly easy for the programmers. So that has been incredibly valuable, it's fantastic. It's the way it should be, you know. If you have the opportunity to work that way with an interaction designer, that's incredibly good."* [D3-560]

All specifications, mockups and prototypes are discussed and validated with the PO and the expert users before development is started. During development, D4 is available for discussions and clarifications, and provides additional details when needed.

D-4 mentions that it is somewhat challenging to know when the specification phase can be said to be done.

> *"What's a bit difficult, is... when are you "done" with the specification work? I've asked that question in many projects. When can* I *tell a project leader or a developer, or the team, that this task here, the specification is done? And.. and it can't.. that is, I may* think *that the specification is done, but that doesn't mean that developers agree."* [D4-179]

5.5.3.2.2 User testing/evaluation

All designs and mockups are continuously evaluated by the expert users as they are produced, but so far there have been no "formal" user tests. Currently, one is being planned:

> *"[...] and soon we're having a real user test in which we tell them to do exactly the same things they do today, but in front of them we'll present a new user interface, or what we call an improved user interface. And then they'll go through some test scenarios, [...] to find out if the changes we've made make the interface as easy to use as before, or easier."* [D4-353]

The test will be performed using a representative selection of end users from the organization.

5.5.3.2.3 Tasks

The team has tried using both the issue tracker system Jira and the more Agile-oriented project management tool called AgileZen. As those systems did not provide the necessary flexibility due to various reasons (technical/organizational), the interaction designer set up a "manual" system in the wiki system called Confluence. All stories along with necessary information are tracked in this system, using color coding to discern stories that are currently finished or unfinished.

For a time, the functional team had their own card wall, but this was exchanged for a column on the main card wall when the project moved into new offices:

> *"Initially the three members of the functional team had their own card wall, and we also had the main card wall of the developers. That was practical, because.. we'd started using the developer card wall and thought it worked very well, with stand-up meetings and so on. There were three members in the functional team back then as well, and they had a lot of tasks, and "who's doing that"? They didn't have control of their tasks, so we made them their own card wall, with all the functional tasks [...] analysis, mockups, screens. [...] So they had their own little card wall, but when we switched offices we figured that.. no, we'll take it in.. we'll gather the tasks into a column, and we'll put it on the main card wall so that everyone works within the same card wall. So [D4] and I spent a lot of time planning the card wall, how to make the flow work. And it made us a tighter unit as a team."* [D1-425]

The Specification column is still in use. Asked whether a task like performing a user test would be represented on the board, the interaction designer responded:

> *"Well, as I've not been part of the project for long, and haven't always been a full-time member.. this is the first user test we'll run in this project, besides the continuous testing performed by the expert users. So.. we haven't really had a card on the wall yet, that reads something like "[D-4] will do a user test". But.. it's very possible that such a card will be put on the wall. But it won't be defined as a separate task in the backlog, I think. It's not a separate backlog task, doing a user test."* [D4-424]

5.5.3.2.4 Role

The interaction designer (D-4) was brought in both in order to improve the user interface of the product, and to help improve the specification-related work processes like writing user stories. He has worked closely with all parts of the project team: the PO/functional team, the project leader, the developers and the expert users.

"An interaction designer, or like I feel I am, really.. you're quite versatile. I talk a lot to the expert users, those who use the system, I have in a way.. when there's a task... a lot is related to user experience, and that means you have to talk to the expert users first, in order to understand what they actually mean. [...] You need to get a good description of that. And I'm the person who talks to both parties, I'm in contact with the expert users, and I'm in contact with the developers. On the same task. [...] And there's a lot [of contact] with the Product Owner. [...] So.. I move around a lot and communicate with a lot of people on the project. But I do feel that the developers communicate well with the expert users too." [D4-471]

Neither the PO nor the developer had previous experience with having a UX professional on the team, but both consider it to be a considerable improvement:

"[...] and before [D-4] came in, we had specified quite a few tasks related to user experience, so in that way we saw the need to.. we knew the system wasn't tip top in that area. But I don't think any of us.. I didn't know any interaction designers.. and.. on other projects I've been on, it's been the last thing anyone thinks about. But I've experienced it as a very positive element." [D2-460]

"Oh, god, yes. Oh.. definitely, everybody respects the interaction designer a lot, there's no doubt about it. It's... yeah. I think all the developers think so, you know, it's like.. it's just so much simpler, everybody wants it to work that way now, you know. Of course, some developers are able to do those things themselves, but we're not that *good at it, we know how to make user interfaces as long as we get the specifications, but when we get them like that, as a prototype you can actually click your way through, see how things work, that's incredibly good. It's very fun to work like that."* [D3-568]

Both the project leader and D-4 himself repeatedly mention that he has been heavily involved in adapting and improving the development process. And although he feels that working on Agile projects can sometimes be challenging, D-4 thinks that the process is working well in this case:

"Yeah.. the developers and I work very well together, I think.. it's always a bit challenging to be an interaction designer in a project like that [Agile], I won't say differently. But of course, generally in Agile, being an interaction designer... but I feel it's working well on this project, actually." [D4-449]

"[...] and we have a pretty open dialogue, expert users and developers and interaction designer, we talk a lot, because we have a goal, we have one common goal.. and it's important that we communicate." [D4-499]

"But on.. other projects I've been on, there's been.. well.. now, *I feel we have a very good process. Of course, there are always rocks in the sea, there always will be, but.. the systematics of Kanban, having everything visualized on the card wall.. seeing that people*

communicate, and knowing that you're communicating with the right person, that's a kind of bonus as well [...] I'm very happy with that." [D4-550]

5.6 CASE E

5.6.1 HOW THE STUDY WAS CONDUCTED

A middle manager/project leader in the organization was contacted by e-mail and asked about the software development process used, the amount of UX resources available in the organization, and whether it would be possible to conduct interview(s) with relevant team members. The manager responded favorably, and an introductory meeting of about 1 hour was held a week later at the organization's premises. The organizational structure, current project(s), the development processes and UX work were discussed. Key points were recorded in field notes. Permission was obtained to contact and interview the resident UX specialist.

The main interview with the UX specialist was conducted about a week later. The whole interview, lasting about three hours, was recorded and soon after transcribed verbatim. An interview guide developed prior to the session was used to ensure that all topics were discussed, although the wording and the order of the questions were changed during the interview.

Additional materials used were printed versions of several presentations previously held by the interviewee, as well as (online) meeting agendas and minutes, lists of reported needs from users and a prioritized backlog.

5.6.2 CONTEXT

5.6.2.1 The organization

The organization is a large institution in the Norwegian public sector, with several thousand employees. The ICT section has about 200 employees organized in several subsections, and a varying number of project- and service groups performing development tasks across sections. The interviewee is the only one currently employed as a UX specialist, although there are some additional resources in the organization with some background in UX (information architecture and design).

5.6.2.2 The project(s)/product(s)

The interviewee is currently mainly working on two interrelated products. One is a Content Management System (CMS) developed from scratch by the organization, which is currently used by several institutions in addition to the organization itself. The other is the main web site of the organization, consisting of several large subsites as well as the main site shared by all. The site is based on the CMS, and the development of both is interrelated. The site has about 1.000 editors (users of the CMS). The development of the site has been organized in a large, "traditional" project that ran for several years, although most of the end result has been produced during the previous two years.

5.6.2.3 The interviewee

E-1 has the equivalent of a master's degree in Information Systems with a specialization in Human-Computer Interaction. He has been working as a usability specialist for about ten years. The last eight years he has been working in a (partly) Agile setting, but he also has prior experience (about two years) with "traditional" processes. He also has some experience with technical development, mainly web-related/front-end.

5.6.3 Relevant findings

5.6.3.1 The development process

The development process used in the organization does not follow a standardized Agile process model, although it does adhere to the definition of Agile by Abrahamsson et al. (2002) (see section 2.1.1), with the possible exception of "easy to learn" and "well documented": according to the interviewee, no process documentation exists, and he explicitly mentions (unprobed) that the process might seem "random" to new team members.

As is often the case when the development organization is an internal department, it is difficult to have a clear-cut definition of the concept "customer". It is perhaps better to use the term "internal stakeholder" – employees (or units) within the main organization that have some form of stake in the product that is being built. Typically they will be end users of the product when it is first released in its usable form, or will be promoting the product to external users. In this case, communication and collaboration with internal stakeholders is frequent, with weekly and monthly meetings in addition to communication "whenever needed".

The general development process is described by the interviewee as highly adaptive. The current release pattern is monthly, although bug fixes are released more frequently when needed. Critical errors can be fixed and put into production "tomorrow":

> "*If it's a critical bug, we'll fix it, and it'll be in production tomorrow. It's not like, "er, no, you'll have to wait.." [...] we're agile enough to be able to release quickly, if there's a need for it.*" [E1-1648].

The monthly release schedule is not completely fixed, the release date can be moved, although efforts are made to ensure that the general cadence does not "slide":

> "*[...] and we move it [the release date] all the time. [...] But we try to avoid sliding. To do that we have to reduce scope.*" [E1-1705].

Weekly status meetings with the PO, E-1, developers and various internal project resources are held as a primary means of coordinating the development effort. During these meetings, developers may assign themselves or be assigned to certain tasks according to field of competence and/or prior knowledge about given parts of the system, and follow-up meetings are planned.

The exact approach to be followed when developing a feature or a new product is determined based on the characteristics of the feature/product, and in some cases even a "waterfall" approach might be used if it is determined to be appropriate. E-1 mentions porting an application from one technology to another, where the original application serves as a complete blueprint for the new, as an example of such a case. In more complex (he uses the word "explorative") situations, other kinds of approaches are used. If the feature to be built is somewhat complex, and/or not well understood by the team, a small group might be assigned to do some initial research, E-1 doing the UI-related part of it. Upon completion of the research work, the members of the group are usually assigned to continue developing the feature.

5.6.3.2 UX work, activities and related themes

5.6.3.2.1 User needs, research and design

E-1 is the sole UX specialist in the ICT section. 10-15% of his time is dedicated to providing UX help and input for any and all projects in the organization. The rest of the time is divided approximately equally between the team developing the CMS, and work related to his role as the main information architect for the institution's web site. The understanding and perceived importance of UX varies in the ICT section, but E-1's manager – who could also be said to be the equivalent of a Product Owner (in Scrum) – has a background in the field of UX, and there are several project managers as well as developers who have a general interest in the field.

There are no explicit rules or guidelines for when and how UX work should happen in relation to development activities in the main projects (CMS/site). E-1 takes part in investigating user needs, does up-front research when needed, specifies requirements, draws user interfaces and performs user testing. It appears that the different activities take place if and when there is a perceived need (by E-1 himself, the PO or team members), and not according to a predefined scheme. E-1 has a high degree of autonomy in his field both regarding what should be done and when it should be done, although the Product Owner might specify tasks and deadlines:

> *"[I am] responsible for [my own tasks]. That is, that I deliver according to my responsibilities. But of course, when [the PO] comes to me and says "you have to fix this" or "you have to take a look at this", it needs to go on my list. And then... it depends on when he expects... if he expects to have it done by some specific time, he needs to tell me when that is, and then I have to plan accordingly."* [E1-1669]

The initial phases of the web project – closely entwined with the development of the CMS itself – were run as in a traditional waterfall project. Needs- and target group analyses were performed:

> "*In the previous years there were needs analyses, focus groups and a lot of things [...] where we dredged up any web-related needs thinkable, and we had a list of hundreds of needs that people wanted.. anything from case management systems to project management tools.. anything that could have the word "web" in it was brought to the table [...] the expectations and ambitions of what was actually to be developed, those were extremely hyped*" [E1-369]

When asked whether the results from these initial phases were used in later work, E-1 responds that the initial activities and the focus groups – which he helped facilitate – were mainly useful for getting to know the stakeholders and the internal workings of the institution, although he mentions that they

> *"[...] could have achieved this by running a far less... not spending two million[20] on it, and using less time to do it"* [E1-9]

A UX agency was also used during the initial phases of the site development, creating Personas and doing overall conceptual work. The consultants worked closely with E-1, the Product Owner and the chief web editor. Conceptual designs were put on the agency's web site for comments, and this yielded about 40-50 responses that E-1 describes as not extremely surprising, but useful nonetheless.

[20]Referring to money, the Norwegian currency NOK

Every sub-unit of the institution has one or two people appointed to the roles of "editor" and "superuser", the first being mainly responsible for content, the other for technical support and lesser technical administration. User needs/requests are gathered and discussed at monthly meetings with the editors and the superusers (separate meetings, although E-1 mentions that they have considered merging them). The requests are prioritized in a weekly meeting with the PO, E-1 and the chief editor as regular participants. When a request is approved, either the PO or E-1 makes sure that what needs to be done is sufficiently well understood and described before a task is created in the issue tracker (Jira):

> *"We need to talk to those who requested the feature in order to get a better understanding of the requirement. And of course we need to talk to our developers, to see how it should be implemented, because.. things we put in Jira, those are supposed to be specified well enough for the developer to understand what needs to be done. We can't just write "fix it", at least not if it's complex And new features, for those there should be some sort of thought-out plan for how it should be solved."* [E1-780]

If needed, some initial user research and conceptual design work is performed before a feature is developed:

> *"[I'll go through the feature] as a concept, what are the most important things, what are the user needs. Then I'll try to understand that, and if I think that I'm not competent enough in that area, I'll have to go out and do some research, look at and do product analysis, what are others doing, right, talk to people, do some research if I think it's needed. And in a hectic work setting, actual user research isn't done enough, er.. as usual, but given that... I think most often we try to focus on, you know, let's solve the important things first. We have a strong focus on that."* [E1-953]

Technical development and UX work might start in parallel if the task is suitable for it:

> *"Sometimes I create visual mockups first, and then it's implemented afterwards. But for the kinds of.. where there's kind of obviously.. lots of technical issues they'll have to handle anyhow, and plan for, and assess, then.. it might start more in parallel."* [E1-1027]

E-1 also mentions that he once had to convince project management for the web site project to give him more time to do up-front work:

> *"So I pressed hard to get... a week or two extra to do conceptual work among other things once, because.. I thought we didn't have a chance, because we needed to involve people.. others at [the institution] when we were working on it [...] so it was like... I had to say, "this won't do"."* [E1-2502]

When describing how a central feature of the CMS was redesigned, E-1 states that the process was driven by user needs, and that it was a more "classic" way of working:

> *"When we redesigned [the feature][...], it was very driven by user needs. We sat down, the PO and me, and it was more of a classic approach, we thought about needs and problems to be solved, prioritized them, sketched it up, received feedback from the superusers, the editors and the developers.. and then we implemented it."* [E1-1052]

In another example he shows how he started a developer off with the technical tasks related to a feature before the UI details were thought out, so that the developer would be able to get started without having to wait for E-1 to finish his work, although he specifies that

> *"If it had been, you know, a large task, where I'd been afraid that he might have taken a very wrong way, er.. I'd thought more about it before I started him off."* [E1-2563]

When a feature is built, the process is described by E-1 as "iterative" (the term is used several times) in the sense that new information or problems discovered during implementation might make it necessary to go back and do some rework:

> *"We're running things in a pretty iterative manner. As usual, you know, when you start implementing, things happen, right. Some assumption isn't correct, and so on. Then you need to adjust. And you need to... we have room for that. So you need to go back, and something that looked simple, like "we'll just do it", and then the developers start implementing it, and he sees that.. this is more complex. [...] Then he'll come to the PO or me, or someone else who created the task in [Jira], and we'll talk about it, and find a solution... or we'll find out that.. for instance, that release next week, we won't be finished in time for that, so we'll just postpone it... that might happen too."* [E1-789]

5.6.3.2.2 User testing/evaluation

As seen in the previous paragraphs, mockups – both wireframes and graphically designed sketches – are used to present new features to stakeholders and receive feedback. However, actual user testing using mockups has not been performed.

Usability tests of the web site are planned by E-1, the PO and the chief web editor in their weekly meeting, in which they also evaluate the results and decide what needs to be done:

> *"[...] we plan usability tests, evaluate the results.. we've had lots of usability tests.. of our web site. And we need to act on the results from those. Otherwise there's no point in having them."* [E1-548]

At several points during the interview, E-1 mentions previous and planned future usability tests and the measures that have been taken as a result of the issues found in tests. The UX agency has been hired to do "traditional" testing of the site several times:

> *"In the two years [of development], we've had four or five big.. relatively big usability tests, at [the agency]. With.. filming and observation room and.. heat maps and eye tracking and everything."* [E1-1901]

The results were perceived as generally useful, and he particularly mentions that after having the developers observe the testing from the observation room, they were strongly motivated to fix a problem that they previously had argued wasn't "all that important".

E-1 has also performed more lightweight tests. At one point, he brought a laptop to one of the central buildings at the institution, and asked random passers-by to perform simple tasks like locating relevant information at the site. Recently he tested parts of the CMS user interface that had shortly before been redesigned, showing up at the office of two of the editors to test "in situ", and using a laptop and a meeting room for two other tests (no special equipment or software).

> *"[We tested] four or five. [...] There was no big report or anything, I just summed up the results quickly. And then we made some changes to the UI after that.. on some terminology and stuff."* [E1-1954]

When describing one of the lightweight tests, he states that there is not enough user testing:

> *"So we've had a lot of usability tests, but in an ideal situation we should of course have done more testing."* [E1-1992]

Later in the interview, he repeats that there is not enough time to do testing. Towards the end, when the subject of Agile (short) time frames comes up, he states the following:

> *"[Within Agile time frames] it can be difficult. But in my experience it's generally difficult to find the time, even if [the process] is as ad hoc as possible."* [E1-2594]

The 10-15% of E-1s time that is set aside for assisting other projects in the organization, is mostly spent doing heuristic evaluations of web projects right before they are launched ("it's firefighting, really"), as well as providing advice for hiring other UX professionals if there is room for it in the project budgets. Even though management has made UX evaluation before launch mandatory for all projects, this is not always done, as not everyone thinks it's necessary.

5.6.3.2.3 Tasks

E-1 largely decides what needs to be done in terms of UX work himself, but the PO might specify certain tasks, and the developers (or other people in the organization partially involved in the projects) might spot usability problems. Developers/others either talk directly with E-1, and/or they may create Jira issues assigned to him. He keeps track of his own tasks separately from development tasks:

> *"I have my own TODO-list and a project list and.. in which I have larger tasks that I'll have to do at some point, and then I have the things I need to do now on a TODO-list [...] I'm responsible for [the list] myself."* [E1-1657]

When asked whether he thinks it's difficult to estimate UX-related tasks, he responds

> *"No.. I don't think it's difficult to estimate tasks related to interaction design and information architecture [...] I think it's easy.. [...] It's like.. it gets as good as... you only have time to do what you have time to do."* [E1-2487]

5.6.3.2.4 Role

E-1s role as UX specialist seems well established in the projects he spends most of his time on. Due to his technical competence and project experience, he is able to communicate well with the developers. At several points in the interview he mentions that technical expertise is an advantage when making design decisions, even though it might be a slight hindrance for creativity:

> *"I was interested in having a bit broader focus, and especially understand the client side, and I felt that it was essential for an interaction designer.. to at least have a.. reasonable understanding, of what is smart, what is possible, what is expensive."* [E1-112]
>
> *"You could probably think that you're... what should I call it,* hindered *in thinking creatively, when you know about all the technical limitations. And it might.. sometimes that might be the case, but I think that overall it's incredibly advantageous to know those limitations."* [E1-1697]
>
> *"If you've got an interaction designer with some technical expertise, you won't completely miss the target. Or.. you know.. I usually know it when I do something that I'll probably get chastised for later, or something that's difficult, I know it, so I do it consciously if I.. if I think.. okay I'll just do it, and we'll discuss it afterwards."* [E1-2457]

When asked whether he thinks that his (UX-related) expertise is respected by the developers, he responds that he thinks so:

> *"My impression is that they think I make good decisions, and that I know what I'm doing. But I'm not infallible, so they have to.. well.. they're not afraid of suggesting things for me, or disagree, I think, or at least they shouldn't be."* [E1-1159]

He also mentions that his decisions are generally respected, although UX-related improvements and goals might not always be prioritized, as the external pressure to deliver functionality is strong.

The CMS/site projects have used a graphic designer hired from the UX agency previously mentioned. E-1 coordinates how the designer's time is spent, keeping a backlog of design issues that they work on collaboratively when he is brought in. The relation between interaction design and graphical design is mentioned and discussed by E-1 at several points during the interview. He mentions that there was a "power struggle" between the two disciplines at a previous workplace, and he is adamant that graphic design should be done *after* the initial planning of the user interface, although it is sometimes necessary to make adjustments to the initial designs when the graphic design is created. However, he is equally adamant in his opinion that graphic design is absolutely necessary for the user experience:

> *"You can't create a good user experience on the web without involving a graphic designer. It's that simple. And those of you who think differently, are wrong, we've tried telling people..."* [E1-1296]

5.7 Cross-case summary

All of the cases describe development processes and UX-related work within large organizations running multi-year projects. Every project employs at least one UX professional. Three of the products being developed are case management systems targeted at an internal audience (cases B, C and D), although the user interfaces are built and presented using web-related technologies. In cases A and E, the products are targeted at an external audience.

In all cases except C, parts of or most of the product is already in regular use, making parts of the development effort maintenance work. In cases A, B and E the actual development project is close to being finished, although a certain amount of development will continue after project closure in addition to regular maintenance.

In cases A and B, the development process is based on Scrum. Modifications to the standard process model have been made in both cases. In case A, two pre-planning meetings have been added to every Sprint in order to support requirements specification work. In case B, several regular meetings have been added in order to overcome size-related challenges, and for a time, UX professionals working on the project held coordination meetings about twice weekly. In both cases, several specialist roles have been added, among them technical architects and UX professionals.

In cases C and D, the development process is based on Kanban. Both apply the two main tools of Kanban, a visualized workflow and work in progress (WIP) limits for certain phases in the flow. In both cases, the resident UX professional has been involved in adapting and evolving the process.

In case E, no specific Agile process model is used, but the process mostly adheres to the definition of Agile found in Abrahamsson et al. (2002). The UX-related work processes are mostly similar to the ones found in the other processes, resembling cases C and D slightly more than A and B.

In all cases, there has been a phase at the beginning of the project in which overall requirements were identified. In all of the cases except E, the UX professionals entered the project after this phase had concluded. In cases B and E, the overall requirements work included UX-related activities like creating Personas and workshops conducted with end users.

All of the UX professionals interviewed are either responsible for, or heavily involved in, transforming overall requirements (or "epics") into user stories that contain enough information for the developers to start their work. In all cases, mockups are used extensively as communication tools, promoting a common understanding between all internal and external stakeholders. In cases A and D, high-fidelity ("clickable") prototypes are often created.

In all cases except E, a physical card wall visualizing the work flow is displayed in the room where the project team is working. In the Kanban cases (C and D), the workflow includes an up-front specification phase, in which information is gathered and mockups created prior to development work. In the Scrum cases (A and B), specification is not part of the workflow mapped out on the card wall.

In all of the cases, most UX-specific tasks like conducting user research, user tests and creating mockups are not tracked on the card wall. In case C, UX tasks are tracked on a separate board. In case D, the same solution was in use for a time but later abandoned. In cases A, B and E, the UX professionals keep their own, separate to-do lists, although UX-related tasks may be added to the electronic issue tracking systems.

A group of expert users is involved in every case. The expert users are highly knowledgeable users chosen to participate in the project, assisting development work by taking part in specification efforts, validating designs and testing working code. In cases B and C, it has sometimes been somewhat difficult to gain access to other, more representative end users.

Most of the UX professionals interviewed mention that there hasn't been as much user testing and/or user involvement as they would have liked. In case A, several "traditional" user tests have been conducted using an external company, but no light-weight/"guerrilla" user testing. In case B, three "mid-weight" tests have been conducted, but testing was stopped as there was no room for acting on the feedback received from the tests. In case C, paper prototype testing has been found highly useful. In case D, a user test is currently being planned on working code, no others have been conducted so far except for the continuous expert user evaluations. In case E, several user tests have been conducted using both "traditional" testing and lightweight testing.

In cases D and E, the UX professional usually works very closely with the Product Owner. In cases A, B, and C, several UX professionals are or have been involved on the same project, and one is usually working more closely with the PO than the others, although all communicate with the PO on a regular basis.

None of the interviewed UX professionals have experienced any (severe) problems working with the developers on their current projects. Several mention that having a technical background and/or understanding is advantageous. Both of the developers interviewed (cases A and D) are very pleased with having a UX professional on the team.

All of the interviewees who have experienced working within "traditional"/waterfall-like projects state that they much prefer Agile projects, regardless of role. However, most of the UX professionals also mention that it is sometimes stressful or challenging to work within Agile projects. In all the cases, UX work is performed both up front, in parallel to and after development work, and most of the effort is usually spent on the specification work. All of the UX professionals are closely co-located with (or even sitting at the same table as) the developers, but are not counted as part of the actual development team, although in case A efforts have recently been made to completely include the UX professional in the development team.

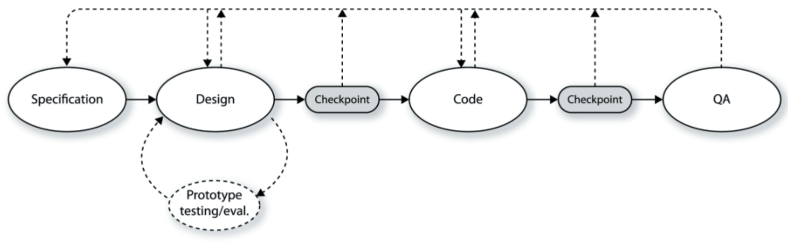


Figure 4: life cycle phases of a user story, descriptive model. The full model may be found in Appendix D.

Although the teams use different processes and time boxes, many of the similarities may be described through a general model of the life cycle phases of a user story. The full model is presented in Appendix D.

The model is merely intended as an aid for understanding, and is not supposed to be prescriptive. The main point of interest is that all of the processes seem to implement some form of checkpoint, formal or informal, before and after the coding phase. If a story does not pass a checkpoint, it is usually moved back to a previous phase for further elaboration or improvement.

6 VALIDITY

In this chapter, issues that might affect the validity of the study are discussed. As interpretivist research needs to be judged according to a different set of criteria than positivist research, an overview of criteria is presented in section 6.1, followed by a discussion of specific issues related to the study in section 6.2.

6.1 JUDGING THE QUALITY OF INTERPRETIVIST RESEARCH

The standard set of quality criteria for judging research – objectivity, reliability, internal and external validity – stem from a positivist worldview, and are thus problematic to use directly in an interpretivist setting. Oates (2006) refers to a set of criteria proposed by Lincoln and Guba (1985) that are an alternative to, but parallel to, those for positivist research, described in Table 6. The criteria have been criticized (even by the original authors) for being an attempt to force interpretivism into a positivist framework, but have been accepted by many (Shenton, 2004).

Positivism	**Interpretivism**	
Validity	Trustworthiness	How much trust can be placed in the research
Objectivity	Confirmability	Whether there is enough information about the study to judge whether the findings do flow from the data and the experiences in the setting
Reliability	Dependability	Whether the research process is recorded and the data documented sufficiently well for others to trace the process
Internal validity	Credibility	Whether the enquiry is carried out in a way that ensures that the subject of the inquiry is accurately identified and described. Can be achieved by triangulation (of data, methods or theories) or by having the informants check that the write-up is correct (respondent/member checking)
External validity	Transferability	Whether the findings may be transferred to other cases, and whether the description is sufficiently "thick" (detailed) so that the readers can judge whether their own situation of interest has similar features

Table 6: criteria for judging quality in positivist and interpretivist research, adapted from Oates (2006)

Oates likens interpretive researchers to lawyers in a court: both "have to make arguments and convince their audience that their descriptions, explanations and interpretations are plausible and supported by evidence (data)" (ibid., p. 295). Interpretive researchers have to explain how the data was gathered, and reflect on how they themselves may have affected the data.

Case studies are often associated with the interpretive paradigm, although they can also be used within the positivist paradigm. In his book on case study research, Yin (2009) mentions some of the criteria in Table 7, but focuses mainly on the traditional ("positivist") criteria – "tests" in his terminology – and tactics for increasing validity and reliability in a case study setting. In the view of the author of this thesis, the tactics mentioned are not in conflict with the Lincoln and Guba criteria. (The tactics listed for internal validity are mainly relevant for explanatory case studies.)

Tests	**Case Study Tactic**
Construct validity	• multiple sources of evidence • establish chain of evidence • have key informants review draft case study report
Internal validity	• do pattern matching • do explanation building • address rival explanation • use logic models
External validity	• use theory in single-case studies • use replication logic in multiple-case studies
Reliability	• use case study protocol • develop case study database

Table 7: Case study tactics for four design tests, adapted from Yin (2009).

Berg (2004) provides additional insights regarding objectivity and generalizability in relation to case studies. If the question of objectivity is seen as the ability of the researcher to articulate the procedures used in the research so that others can repeat the research if they so choose, case studies may be seen as no different than other data-collection and analysis strategies used by social scientists. If the findings are doubted, anyone is free to replicate the research with a similar case subject. Subsequent research will then either corroborate the original findings, or show that those were faulty, in error, or inaccurate.

Within the interpretivist paradigm, the question of generalizability is not necessarily a meaningful one, since there is clearly a scientific value to gain from investigating an entity simply in order to gain an understanding of it. However, seen from a positivist viewpoint case methods are still useful, and to some extent generalizable, owing to characteristics of human behavior: “When case studies are properly undertaken, they should not only fit the specific individual, group, or event studied but also generally provide understanding about similar individuals, groups, and events. [...] The logic behind this has to do with the fact that few human behaviors are unique, idiosyncratic, and spontaneous. In fact, if this were the case, the attempt to undertake any type of survey research on an aggregate group would be useless.” (ibid., p. 259).

6.2 VALIDITY ISSUES

The overall concept of validity is paralleled by the concept of “trustworthiness” in qualitative research. In this section, issues affecting the trustworthiness of the present study are discussed in relation to the four quality criteria presented in the previous section.

6.2.1 TRANSFERABILITY (EXTERNAL VALIDITY) ISSUES

The most important threat to the transferability – the interpretivist parallel to external validity – of the results from the present study, is the selection of cases. Yin (2009) recommends that the cases in multiple-case study designs should be chosen according to a replication logic, not a sampling logic. Four of the cases were chosen based on a literal replication strategy (two Scrum cases, two Kanban cases), and the fifth (case E) was included based on a theoretical replication strategy (to look for contrasting results, in this case in a situation with limited UX resources on the project(s) compared to the others). However, the cases were also chosen based on convenience, and due to certain similarities between the projects, the transferability to other types of settings is questionable: all of the cases describe Agile/UX integration in mature teams on multi-year projects in large organizations, and all of the interviewees are either highly educated, highly experienced, or both. Also, the limited number of cases, combined with the very limited geographical spread, is of course a threat to transferability in itself.

Still, the main results largely adhere to those of previous studies (see section 3.2.2), and from the literature review it is clear that most teams experience the same kinds of challenges and benefits[21] across different settings. Also, the detailed description of results

[21]It should be noted, though, that there seems to be a certain bias in the reviewed literature. All report on some degree of success regarding the integration, although several blog posts, forum posts and other non-peer-reviewed sources indicate that success is not always had (one example is found at http://webtorque.org/?p=1060). This bias is perhaps not very surprising, as anyone presenting a "failure story" risks that the audience might attribute the failure to the

(chapter 5) should provide both practitioners and academics with the opportunity to decide which parts of the findings may be relevant to their purpose.

6.2.2 *Credibility (internal validity) issues*

One of the main threats to the credibility of the study is the heavy reliance on interviews as a primary data source, especially in the cases where only one person was interviewed. What people say they do and what they actually do (or did) may often differ for many reasons, and the focus and apparent interest of the interviewer may also affect what is said. The main strategy used to increase credibility was to return complete write-ups of the cases to all participants, and ask for comments and corrections. Six participants provided feedback, and adjustments were made according to this.

All interviewees were given a short description of the general subject of the thesis (UX/Agile integration) at the beginning of the interview, with the added information that the interviewer/author has worked both as a (mainly front-end) developer and as an interaction designer. This might have affected the willingness of the interviewees to impart negative experiences or attitudes regarding UX professionals or work practices in general. However, in the experience of the author, the information had more of a positive impact, as the subjects seemed to be comfortable discussing certain problems knowing that the interviewee had personal experience in the field. Several asked questions about the interviewer's personal experiences ("have you found a way to do that?"), but mostly after the actual interview had concluded.

While most of the topics discussed are seemingly non-controversial, they might not always be experienced as such. The ever-ongoing debate on Agile processes in the professional community can in some ways be said to be characterized by strong and polarized opinions. Many feel that there is only one "right" way to do Scrum or other Agile processes, and there has been a lot of focus on so-called "anti-patterns". In some of the interviews, it was clear that the interviewee felt the need to defend process-related choices and adaptations that had been made, repeating statements like "I know it's not by the book, but..." and "this might not seem like Agile, but...". This may possibly have affected process-related details, although in most cases the details could be confirmed by other interviewees or documents. It should also be noted that in both of the Kanban cases, the process has been in use for a relatively short time, and one or several of the interviewees have been involved in setting up the

people/project/organization implementing the process, and not to attributes or characteristics of the process model itself.

process. As Patton (2009) writes, "there's no one as zealous as the newly converted", and this should be kept in mind while interpreting the results, even though two of the interviewees mentioned that they were aware of this source of potential bias themselves.

In all the cases, the project has been going on for several years. It is always difficult to remember what happened several years ago, especially when teams and processes have changed a lot over the years. Human memory tends to be biased toward positive experiences[22], and so problems and challenges experienced in the past may not have been given as much attention as they would have had if the interview was conducted at the time. Also, although the interviewees knew the results would be anonymized, it is usually difficult to discuss problems that are thought to result from personality traits or team characteristics, even though they might very well be common problems experienced by many in the same kind of situation.

Although all agreed to have the interview recorded, and all were assured that the audio files would only be used by the researcher(s), it is obvious that most speak a little less freely than they would have done in a normal conversation with the researcher. Almost all of the interviewees appeared to forget about the recording after a while, but in hindsight – listening to the transcripts – two seemed somewhat uncomfortable, and use of the recording device should probably have been stopped.

6.2.3 Dependability and Confirmability

Both Oates (2006) and Shenton (2004) highlight the importance of providing detailed descriptions of the research design and how it was implemented, in order to address the issues of both dependability (the parallel to reliability) and confirmability (the parallel to objectivity). Regarding dependability, other researchers should be able to repeat the work, if not necessarily to gain the same results. A description of the overall study design and implementation is found in chapter 4, and additional details are provided for each case.

Regarding confirmability, Shenton also states that "Here steps must be taken to help ensure as far as possible that the work's findings are the result of the experiences and ideas of the informants, rather than the characteristics and preferences of the researcher." (ibid., p. 72), and that a key criterion for confirmability is the extent to which the researcher admits his or her own predispositions. These may largely be derived from the introductions and

[22]http://psycnet.apa.org/index.cfm?fa=buy.optionToBuy&id=2003-00781-006

explanations throughout the thesis, and the propositions presented in section 4.2.2 provide an overview of the theoretical basis used in the study.

The author's personal experience in the field has in many ways affected the whole research process, from study design to analysis. Any kind of qualitative research is to a certain degree interpretative, and the author's previous knowledge, beliefs and attitudes affect the results. In the interviews, this effect was sought countered by letting the interviewees speak as freely as possible, using mostly "verbal nudges" ("okay?", "how is that usually done?" etc.) and affirmation ("I know what you mean.."). Transcribing the interviews verbatim and coding the content according to a predefined, common scheme helped avoid issues of interpretative bias that would probably have affected the results more strongly if only notes had been taken, and having the participants review the draft case study reports has helped avoid misinterpretations as well as exaggerating the importance of certain elements compared to others. The results have also been continuously discussed with other practitioners (both UX specialists and developers) to help avoid issues of personal bias.

7 Discussion

In this chapter, results from the case studies described in chapter 5 are discussed in light of the propositions presented in section 4.2.2. In section 7.2, some thoughts on the emergent theme of teams switching from Scrum to Kanban are discussed.

7.1 Cross-case discussion of propositions

7.1.1 I) Integration effort

Original proposition: "*As Agile software development processes are mainly developer- and code-oriented, it requires conscious and focused effort to effectively integrate UX-related work practices and processes.*"

As seen in chapter 2 and 3, many have found that Agile processes lack rules and guidelines for UX-related work (section 3.2.1.1), but also that the two can be successfully integrated (section 3.2.1.2 and 3.2.2). The list of common problems and solutions found in Miller and Sy (2009), as well as most of the papers mentioned in chapter 3, more than suggests that many UX professionals struggle when trying to adapt to Agile processes – but also that the community as a whole has found many viable approaches and techniques to overcome the inherent difficulties.

In all of the cases in the present study, the UX specialists have adapted their work processes to fit the Agile pattern of incremental development and frequent releases. Requirements specifications and interface designs are created shortly before development in a just-in-time manner, and developed features are validated and tested shortly after having been finished. However, in cases A, C and D, it is clear that the development process has also been adapted to fit UX work processes. In case A, two pre-planning meetings have been instated during the Sprint to support specification efforts. In the Kanban cases (C and D), a short up-front specifications phase has been added to the general workflow. In case E, the adaptations are less clear-cut, but it appears that both the development process and the UX work process have largely evolved together, each continuously adapting to the other over the years. In all of the cases, both UX specialists and developers validate their work products with each other before and after development, respectively.

In case B, it appears that the development process was not sufficiently adapted in order to gain full advantage of the UX specialists' work. Although the UX specialists took part in specification efforts, the pressure to deliver quantity rather than quality (in UX terms) reduced the design efforts to "choosing the least bad solutions", as mentioned by B-1. Also, the team was not able to prioritize improvements to existing functionality based on feedback from usability tests, and so testing efforts ceased.

Although adaptations have been made on both sides, it is clear that one problem is felt by all of the UX specialists: the lack of possibilities for going back and redoing features that have been developed previously. This is further discussed in section 7.1.5.

7.1.2 II) Parallel track

Original proposition: "*A common pattern of integration is the one called "parallel track", as described by Silva et.al. (2011). This form of integration would be expected in cases where there is a) one or more dedicated UX specialists on the team, and/or b) where the Agile process in use is relying on specific time boxes (e.g., "Sprints" in Scrum) for development, and/or c) where UX has an established, valued position in the development organization.*"

The parallel track approach described in section 3.2.1.5 might seem self-evident to many UX professionals who have worked on Agile teams, based on the notion that designs need to be created and validated before they are developed, and that testing must necessarily happen after the functionality has been developed. However, both of these notions could be challenged: designs could be created collaboratively with the developers (using techniques like the Design Studio described by Ungar and White (2008)), and/or within the same Sprint (as the team in Hodgetts (2005) tried and rejected), or the UI could be implemented directly and later iterated upon, without using mockups and prototypes as an intermediary stage between requirement and implementation. User testing/validation might also be performed immediately whenever a story is finished, in larger chunks whenever a testable portion of the product is finished, or never (as is often the case in practice).

Also, there is of course the approach that avoids actual *integration* altogether: all UX work is performed up-front in an early phase in the project. The overall concept and designs are then handed off to a development team using some Agile flavor, and the complete product is then tested (if at all) when all development work is finished. Many practitioners working in a web-related setting will recognize this approach – four of the UX practitioners interviewed (B-1, C-1, D-4, E-1) mentioned that they have worked in this manner on previous projects, and a variation on this theme is seen in Ferreira et al. (2011, 2010). At the opposite end of the scale, the "Generalist" approach as described by Fox et al. (2008) can be found, in which every team member is equally responsible for UX work, and UX tasks are treated in the same way as any kind of development work.

In a very broad sense, all of the cases may be seen as using the parallel track approach, with the UX specialist(s) working in a separate track that runs in parallel with the developers'. User research is ongoing, and usually happens at least one or more sprints (in the cases where such are used) ahead of development. Design work is also done in "chunks" prior to development and fed to the development team, usually shortly before development starts. During development, UX specialists and developers communicate directly whenever needed, and provide mutual feedback.

However, regular evaluation/validation is often performed by the expert users, the Product Owner and/or the UX specialist *before* the Sprint/cycle ends or code is deployed to production. Usability testing on non-expert users is performed at irregular intervals (see section 7.1.5), cadence not guided by the Sprints. Also, although Sprints are not used in cases C, D and E, all use a regular pattern of releases/deployments which can be thought of as "cycles" or "iterations" (D and E release monthly, C more irregularly but have kept the two-week intervals between reviews from Scrum). In these cases, research and design may be done both in cycles *prior* to the cycle in which the feature is developed/coded, and *within* the same cycle.

It should be noted that several interviewees expressed opinions along the lines that Scrum feels too inflexible, mainly because of the rigid time frames (B-1 and C-1 make a specific point of this). It is not always easy, or even possible, to divide all kinds of work into chunks that fit the allotted 14- or 30-day time frames. In case C, Kanban was chosen explicitly to improve the specification-related parts of the process and to make room for more collaboration within the project team. In case D, the resident UX specialist was heavily involved in setting up the process. In both cases, the interviewees feel that the new process is working very well, and the UX specialists feel more like actual team members instead of being "a bit on the side". Some related thoughts on this matter are presented in section 7.2.

When the time frames involved are flexible rather than fixed, and extensive collaboration between developers and designers is desired, it may be more fruitful for process designers to focus on (all of) the work and phases involved in developing a single user story, and design for a continuous flow combined with mechanisms to ensure that all roles are involved in the appropriate phases and decision points. A *descriptive* model of such a flow is presented in Appendix D. To arrive at a *prescriptive* model, significant amounts of additional research and analysis is needed.

7.1.3 III) Tasks

Original proposition: "*UX tasks and activities are usually recorded, discussed, estimated and tracked separately from technical tasks.*"

From the papers reviewed in chapter 3, it appears that this specific area has not been given particular attention in previous empirical studies. The proposition was based on the hypothesis that valuable information about UX/Agile integration can be gained through a comparison of how UX and development tasks are tracked throughout the process. As mentioned in section 2.3.2.2, the Agile ideals of shared responsibility and overlapping skill sets should make every task, topic and issue relevant for every team member. For teams working very closely with the UX specialist(s) in a collaborative manner, or for teams completely integrating the UX specialist(s) as a full member of the team, it would be expected to find that UX-related tasks were treated much the same way as development tasks: recorded and tracked in the same system, discussed and estimated at the same meetings. At the opposite end, a low degree of cooperation and collaboration between developers and UX specialists would in all likelihood predict a separation of the two.

It should be noted that the results largely depend on how the concept "UX-related tasks" is defined. In the interviews, this was exemplified as "gathering user information, creating designs, mockups or prototypes, performing usability tests etc." when the interviewees were asked about the topic.

A somewhat unexpected similarity between the cases was that all the teams except in case D used the issue tracker system Jira[23], and that all except case E use a physical board/card wall as well, finding the latter highly useful as a coordination and communication tool. In the Scrum cases (A and B) and case E, some or most tasks were tracked in separate lists by the UX specialists, while some tasks might be assigned to them in Jira. Some UI-related issues might be discussed at regular team meetings, while some would be discussed in separate meetings with other specialists, expert users and/or the PO.

In both the Kanban cases (C and D), UX-related tasks are or have been tracked on a separate card wall. However, the separate card wall strategy was abandoned by D, and a Specifications phase was added to the main card wall instead. In case C, the interviewee stresses the importance of the fact that the separate card wall is only for coordination

[23]http://www.atlassian.com/software/jira/overview. Jira was also used for a time by the team in case D, but was replaced by the wiki system Confluence.

purposes between the UX specialists and the PO, and that the "developer" card wall is the *main* focus of everyone. In both, the whole team including the UX specialists and the PO attend the regular meetings.

Cases A, B, C and D all use the user story format for requirements, and these are broken down into smaller tasks before development begins. It appears that generally, tasks related to UI *design* are usually tracked and often discussed in the same systems and meetings as development tasks. Tasks related to user research and usability testing are usually tracked and discussed separately. This might imply that UI design is regarded as part of the "actual" development phase, while other UX-related tasks are mainly regarded as belonging to prior and subsequent phases (analysis, evaluation), thus not being *directly* relevant for the development team.

7.1.4 IV) Up-front design

Original proposition: "*The amount of up-front design work done in an integrated environment varies. For products that are more information-heavy than interaction-heavy, more up-front work can be considered necessary.*"

From the papers and empirical studies described in sections 3.2.1.6 and 3.2.2, it is clear that *some* amount of up-front design work – in this context, UI design and detailed specification *before* development starts – is usually always done, even in teams using the Generalist approach (as in Fox et al. (2008)). This is also done in all of the cases in the present study. Unfortunately, there is not sufficient data to be able to *compare* the amounts of up-front work between cases. When asked about how much time is used for these activities, the interviewees' responses invariably include "it varies" in all the cases. A slight, overall pattern can be detected in that more up-front work is required in the initial phases of the project, which is only natural as there will often be significant reuse of design and interaction elements (buttons, panels, navigational structure etc.) in later phases.

Measuring the time used for up-front design and related activities would probably be an interesting study in itself, particularly if it was possible to compare cases where "standard" web technologies were used for the interface (xhtml/css/js), vs. more proprietary technologies (Flash). This would of course require accurate definitions of "up-front work", and extensive use of either direct observation and/or detailed time sheets acquired from the project management systems (if such are used).

However, another interesting theme related to up-front design emerged during the study. It appears that in all the cases where requirements are specified using the User Story format, the UX specialist(s) were either responsible for (cases C and D), or heavily involved in (cases A and B), writing the stories. This is in line with the suggestion by Kollmann et al. (2009) that UX practitioners should participate in generating and prioritizing user stories, although prioritization seems to be mainly done by the PO and other stakeholders.

In C and D, it was mentioned that the reason for the UX specialists writing the stories was that the specialists were the only ones who were experienced enough to know how to write the stories and the acceptance criteria sufficiently well. It should also be noted that in several cases, designs were validated (feasibility check) and feedback given to the designers by either a subset or the whole team of developers *before* the designs were considered "done". In cases C, D and E it was also pointed out that designers and developers usually conduct a short "sit-down" meeting when the developers start work on a task, in order to be sure that everyone (still) agrees on what is to be done.

Another perspective that should be mentioned in this context is that of project/product planning. In a sense, all of the projects might be said to be hybrids of traditional and Agile. All projects had a long up-front planning phase (long compared to having *one* "cycle 0" up-front to do planning) in which overall requirements were stated, and those were handed off for the team(s) to detail and develop incrementally. It appears that UX specialists were usually not involved in the project planning phase. This is partly seen as problematic from a UX standpoint, especially if overall product strategies are planned without sufficient analysis and interpretation of user needs.

7.1.5 V) User testing

Original proposition: "*The short time frames in Agile software processes make it difficult to perform enough user testing on actual end users.*"

Interestingly and somewhat unexpectedly, in *all* of the cases a panel of expert users works closely with or within the project team, assisting in design and validation activities. In cases A-D, the expert users also test developed functionality according to specific acceptance criteria in the user stories. In all the cases the arrangement appears to work well, although a caveat should be noted: having expert users work on a regular basis with or within the project team might make it more difficult to gain access to other end users, as in case B. Since the expert users are usually chosen for their expertise, and they often assist in design activities, they will probably not be representative enough for "real" end users when performing usability tests (or other kinds of user research where it is important that such are used).

Apart from the expert user testing, testing on representative end users at *regular* intervals was not found in any of the cases. Somewhat surprisingly, traditional lab testing has been used in all cases except D. Three of the interviewed UX specialists (case A, C and E) mention that there hasn't been enough user testing, and in case A this was specifically attributed to the pressure to deliver functionality. E-1 stated that he feels that it is generally difficult to find the time, regardless of process. In case B, the UX specialist(s) had enough time, but B-1 states that it would have been more difficult if there had been only one specialist rather than three.

In case B, three user tests were performed, but the practice was stopped as the UX specialists experienced it to be pointless: there was no room for changing functionality that had already been implemented. This is reflected in case C, in which paper prototyping is preferred as this only affects the design phase – changing implemented functionality is difficult at best. In case A, several interviewees mention that they have some "design debt", as there is no room for redesigning parts of the site that have low priority whenever new design concepts are created.

It appears that the main problem regarding user testing is not short time frames, but rather that there is seldom room for redoing functionality that has already been delivered/deployed. While Agile processes in general are supposed to be *iterative* – actually *redoing* parts of the product to improve it based on feedback from previous rounds – in reality, the development effort is often simply *incremental*, and new parts are added without ever revisiting the old. (This problem affects code as well as the UI, and is often discussed online by professionals.[24]) Keeping the "incremental" part while not including the "iterative" part of IID processes (see section 2.1.1.1) defeats one of the key advantages of Agile processes compared to the more waterfall-like, traditional ones: the ability to go back and correct previous mistakes.[25]

[24]Examples can be found at http://www.agileproductdesign.com/blog/dont_know_what_i_want.html (Jeff Patton) and http://alistair.cockburn.us/Incremental+versus+iterative+development (Alistair Cockburn).

[25]Many have experienced Agile processes as "a series of mini-waterfalls", as a simple Google search for this expression will show. This term was also used by the interviewee A-4.

7.1.6 VI) *Facilitator role*

Original proposition: "*The UX specialist role will often evolve into a facilitator role in the project, coordinating customer-oriented activities as well as "driving" parts of the development effort.*"

Kollmann et al. (2009) suggests that "UX practitioners need to become design facilitators and should participate in generating and prioritising features and user stories." (ibid., p. 17). This view is reflected in the web article "Twelve emerging best practices for adding UX work to Agile development" by Patton (2008b), in which it is also stated that UX practitioners should be part of the customer or Product Owner team. The results in the present study suggest that this is often the case. In all the cases, (one or more of) the UX specialist(s) work closely with the PO (team), and are involved in generating user stories and specifications.

In the reviewed papers, several practitioners describe how UX specialists may become *design facilitators* rather than "specification owners" (Cho, 2009, Illmensee and Muff, 2009). Rather than having the UX specialists create more or less complete UI designs up-front followed by the developers reviewing them, approaches such as the "Design Studio" described in Ungar (2008) are used: after some limited initial user research, designers, developers and – if possible – representative end users gather to create UI designs collaboratively. This approach does not appear to be in regular use in any of the present cases. Although designs issues may be discussed in groups before mockups are created, the design work is largely done by the UX specialists.

7.1.7 VII) *Cooperation*

Original proposition: "*UX professionals generally work well with developers on Agile projects, particularly when they have a technical background and/or understanding.*"

As mentioned in section 3.2.1.7, some studies have found a certain division or barrier between designers and developers. This topic has been debated in online discussion groups and blogs targeted at professionals for a long time, and the title of one of the courses held at the 2011 ACM CHI conference – "Experiencing Agile Usability: Breaking through the 'Us vs. Them' Problem" – suggests that the division is still experienced by many.

However, in both Fox et al. (2008) and Hussain et al. (2009b, 2009c), study participants find that integrating UX practices has added value to their development process. In Fox et al.

(2008), the authors pose a question in the section "Future work": "An interesting question is exactly how the participants found the addition [of UCD to the development process] valuable?" (ibid., p. 72). At least one aspect of the answer may possibly be found in the cases of this study. Both the developers interviewed (cases A and D) repeatedly mention that they are very happy to have the interaction designer create the UI designs, so that *they won't have to*. (The same perspective is mentioned by B-1.) Both the developers also feel that the product as a whole is considerably improved when UX specialists are involved.

While some of the interviewed UX specialists have experienced minor problems in this area on previous projects, all of them currently feel that there is mutual respect between designers and developers. In all cases, efforts have been made to improve communication and collaboration. It is especially interesting to note that in both the Kanban cases, including the *whole* workflow – including the up-front specification/design phase – on the card wall appears to increase "team feeling", and the feeling among all team members that everyone is working towards a common goal. In both cases, the UX specialists have been heavily involved in designing and adapting the process.

Several of the interviewed UX specialists mention that they think it is advantageous to have a degree of technical understanding, as this improves communication with developers and ensures that designs are feasible. It should be mentioned that this has also been (sometimes hotly) debated in the community of designers working with web-related products for some time[26], and it seems that the opposite view – valuing separation, as the organization in Ferreira et al. (2011) (section 3.2.2.1.1) – does not hold much sway. In the personal experience of the author, technical understanding is invaluable, and it saves both time and misunderstandings throughout the process.

7.2 From Scrum to Kanban

A particularly interesting aspect that emerged during the study was the move from Scrum to Kanban. In cases C and D, the teams had been using Scrum for an extended period of time before deciding to switch to Kanban, and the very mature Scrum team in case A considered switching some time in the near future. For cases C and D, the UX specialists were heavily involved in designing the new process.

Although no actual explanatory attempt will be made here, a few thoughts on this aspect of the study might be useful in relation to future work. Currently, Kanban seems to be gaining

[26]An example of a discussion thread on this topic may be found at http://www.ixda.org/node/30334

popularity at speed. In Anderson (2010), the process is presented as well suited to settings where there is a constant influx of tasks of different types, and where some task categories need to be treated differently. Most practitioners will recognize this situation; when the product being (further) developed is already in use, as in most of the present cases, there will always be an incoming stream of bug reports, crashes, feature requests, and maintenance tasks, in addition to the regular tasks related to the current increment being developed. If there is only one – or few – teams handling both operations/maintenance and development, the time boxes and rules of Scrum may soon become an issue. In rigid and inflexible implementations, urgent matters may be forced to wait for the next Sprint. After a while, the team might get a reputation for being "slow", as important and/or simple tasks appear to people outside the team to take weeks to complete.

A common response to this problem appears to be to shorten the time boxes – shorter sprints make it easier to respond faster. As pointed out by Patton (2009), the user stories shrink as well in order to fit the shorter time frames, which creates its own set of problems. And "short time-boxes herniate" (ibid.) – story elaboration and testing pop out of the box, and the actual time from start to finish may soon become 3-4 times longer than the time box. This is pretty well illustrated by the parallel track model (Appendix C): specification/design, coding and testing take 3 cycles rather than one. After a while, the time-boxes may seem not only rigid and inflexible, but also *meaningless* – after all, when the original point of creating a "potentially shippable increment" *within* the Sprint is gone, organizing the development work in 1-3 week intervals and scheduling planning and review meetings "no matter what" start to look more like blindly following rules rather than actually being *agile*. Two of the participants (B-1 and C-1) express views of Scrum that appear to be related to this issue.

Any development process may be seen as an interface to surrounding processes of different levels and scopes, implementing mechanisms to support and/or plan activities that depend upon, or are related to, the development activities. From the original Scrum literature (Schwaber and Beedle, 2002, Schwaber, 2004), it is clear that the time boxes were intended to improve the interface to business-level processes. Fixed sprint lengths, meetings held at the same time and place at fixed intervals, and the development team's promise to deliver a given set of functionality at the end of each interval, made it easier for management and other stakeholders to plan *their* activities accordingly. However, as the complexity and the importance of the products being developed increase, stronger involvement from the business side is required. When teams collaborate and communicate with the business side continuously, the need for fixed time boxes, meetings and "promises" may decrease. All of the present cases describe mature teams, and the Product Owners and other stakeholders are continuously involved in the development effort. This allows for a more flexible process[27], which may work very well provided that it still provides an adequate interface to related processes[28].

It might be argued that Scrum may be tailored to fit the setting just as well as any Kanban implementation. In the view of this author, the advantage of Kanban from a UX perspective seems to be largely related to its somewhat agnostic approach to process level and scope, and to the strong focus on the *need* for tailoring the process. When UX specialists are involved in designing the process, as in cases C and D, there appears to be a strong chance that the specifications phase will be included in the visualized workflow, along with quality assurance and other related activities. When *all* activities needed for getting an *actual* potentially shippable product increment from start to finish are visualized in the same flow, it is natural to expect what is seen in the cases: a stronger team feeling and sense of shared responsibility, which is precisely what was originally intended in Scrum – but this time including the *whole* project team.

Hopefully, the increased popularity of Kanban will lead to more organizations reflecting on their development processes, and how the quality of the product (in terms of UX and otherwise) is affected by the process-related choices that are made. Software development is an extremely complex activity, and it is made even more complex by the increasing demands on user experience. Some of the complexity needs to be handled by developing incrementally, some of it by *iterating* on both designs and code – and most of all, by constant communication and collaboration between all stakeholders.

[27] In case B, the size of the project and the number of teams (11) suggest that the coordination mechanisms need to be stronger than in the other projects, and indeed they appear to be.
[28] An interesting discussion related to this is found in chapter 11 in Anderson (2010), "Establishing Service Level Agreements".

8 CONCLUSIONS AND FUTURE WORK

8.1 RESULTS, FIRST RESEARCH QUESTION

Two research questions were sought answered in this study. As the field of Agile/UX integration is limited, the first question was formulated using general terms, in order to "cast a wide net" and gather information from a broad(er) range of subjects:

> *1. How can UX-related work practices and processes be integrated with Agile software processes?*

To answer this, relevant literature was searched in order to identify ways in which UX work practices and processes can be integrated with Agile software development processes. The results are described in chapter 3, and may be summarized as follows:

- Agile software processes in general do not provide rules or guidelines as to how and when UX-related work should be performed, or how the role of UX specialist/designer can or should work in relation to the development team.
- Suggested approaches to Agile/UX integration mostly follow the "parallel track" model, as described by several (section 3.2.1.5). Developers and UX specialists work in separate, parallel tracks. Designs and specifications are created one or more cycles/sprints ahead of development, and completed features are validated and tested one or more cycles/sprints after development. Most advocate using a short, up-front analysis/design phase ("cycle/sprint 0"), in which user research and holistic planning of the user interface is performed.
- Effective use of the parallel track model requires using "lightweight" UX techniques, mostly being less resource-demanding versions of "traditional" UX techniques. Examples include paper/low-fi prototype testing, RITE testing, lightweight Personas and informal cognitive walkthroughs.
- Although many have experienced challenges related to the integration of UX work practices and processes and Agile processes, there is considerable agreement that integration is possible.

8.2 RESULTS, SECOND RESEARCH QUESTION

Although a body of work now exists in the field of Agile/UX integration, there is still a need for more empirical research in order to be able to provide advice and suggestions for practitioners (Silva et al., 2011). Also, both fields are continuously evolving, and the practices of yesterday are not necessarily the practices of today or tomorrow. Thus, an empirical study was conducted in order to answer the second research question:

> *2. How are UX processes and UX-related work practices integrated with the Agile processes Scrum and Kanban in practice [in a web-related environment]?*

The goal was to produce detailed, descriptive information about how Agile/UX integration is actually practiced in projects where the development process can be considered Agile, and one or more UX specialists are involved. The Kanban cases were of special interest, as no empirical studies of Agile/UX integration on Kanban projects were found in the search for related literature. The results are described in chapter 5, and the main findings can be summarized as follows:

- All of the projects employed a long, traditional "up front" phase at the start of the project, in which overall requirements were gathered. However, most of the UX specialists entered the project after the up-front phase was finished.

- UX work processes and practices were adapted to the development processes, but in every case the opposite was also true. Meetings, phases and artifacts were added to the development process in order to support UX work.

- The UX specialists usually worked closely with the Product Owner, and were either responsible for, or heavily involved in, writing requirements specifications in the form of user stories.

- Although the UX specialists usually worked in parallel with the developers, creating designs and specifications shortly ahead of development, the "parallel track" model can only be said to have been followed in a broad sense. In the teams using Kanban, the model was less suitable for describing the integration than in the teams using Scrum. As the model is based on time-boxed processes like Scrum, this is to be expected. It is suggested that future work in this area may benefit from using additional, alternative models to describe and study integration.

- In every case, a group of expert users were part of or worked closely with the project team, assisting in design and validation activities.

- Development usually happens incrementally, and most of the UX specialists experienced that once a piece of functionality had been finished and deployed to production, it was very difficult to get the time and resources to make changes to it later on. Iteration thus mainly happened at the design stage prior to development, using prototypes.

- There was mutual respect between developers and UX specialists in every case. In the Kanban cases, the UX specialists felt more like full members of the team. This appears to result partly from adding UX-related phases to the overall workflow on the card wall.

- In four of the cases, a physical board/card wall was used to track progress. This was considered a highly useful communication and collaboration tool by all participants.

8.3 General advice for Agile process designers and project management

All research in this area ultimately aims to provide advice for practitioners, supporting the continuous work of improving processes in order to develop better software in more efficient manners. Practitioners might be interested in reading the full case reports in chapter 5 in order to compare the described practices to their own, and evaluate the possibilities for incorporating adaptations that seem useful. Some general advice is provided in the text box on the next page, based on study results.

General advice for Agile process designers and project management

- ✓ Process designers need to be aware that Agile processes do not inherently provide guidelines for UX roles or UX-related activities. If UX is considered important, and/or UX specialists are part of the project team, the development process should be analyzed and adapted in order to make full use of the specialists' competence and the results of UX-related work such as user research and testing.
- ✓ Agile processes run the risk of becoming a "series of mini-waterfalls", building the whole product increment by increment, but never actually allowing the team to revisit previously developed features to improve the quality based on new information or feedback. The value of improving quality in terms of user experience should be estimated and weighed against the value of providing more functionality, and resources prioritized accordingly. Ideally, the process should have a built-in loop at regular intervals for gathering feedback *and* for acting on the results.
- ✓ Developers and designers should be co-located, and communication throughout the whole process encouraged. Formalized "checkpoints" may also be added to three points in the process in order to improve efficiency: 1) designers validate designs with developers before the designs are accepted by other stakeholders, 2) when development work starts on a feature, designers and developers meet to check that all (still) agree on what is to be done, and 3) developers validate completed features with the designers before considering them to be "done".
- ✓ Tracking user stories in progress on a physical card wall supports communication and collaboration for Agile teams. For users of the Kanban process, including UX-related phases in the visualized workflow is recommended to promote the experience of the whole team working towards one common goal.
- ✓ Project management/process designers should also be aware that having a team of expert users on the project team does not (necessarily) void the need for involving other users in UX-related work.

8.4 FUTURE WORK

The Agile/UX integration field of study is very young, and most aspects are still not very well covered and would benefit from further investigation. However, there is a need for a establishing a common terminology, a set of definitions, and a generally accepted conceptual framework, most importantly in order to support and simplify discussion and comparison between studies.

An obvious issue that should be further addressed is that of developing a more suitable model for describing and guiding integration than the “parallel track” model, for settings and processes that do not rely on fixed time boxes for development. A suggested perspective is that of focusing on the work and phases involved in developing a user story, and designing for a continuous flow combined with mechanisms to ensure that all roles are involved in the appropriate phases and decision points (as mentioned in section 7.1.2). The case descriptions in the present study as well as the descriptive model presented in Appendix D may provide useful input for further work in this area.

A point of interest that has apparently not been covered in the related literature, is how the “lightweight” UX practices compare to their “heavy” counterparts in terms of results. Are the lightweight practices always sufficient for creating quality user experiences within Agile projects? If not, in which settings are the heavier practices more suitable or needed?

Also, a subject that has been discussed for some time in the professional community, but is not covered in the related literature or in the present study, is that Agile (web) projects seem to need more UX resources than traditional, “waterfall-like” projects. A study focusing on this topic would be highly interesting. On a related note, comparing the type and amount of up-front work needed for interaction-heavy vs. information-heavy systems would also provide useful input for organizations planning and staffing new projects.

BIBLIOGRAPHY

ABRAHAMSSON, P., SALO, O., RONKAINEN, J. & WARSTA, J. 2002. Agile software development methods: Review and analysis. Espoo, Finland: VTT.

ADIKARI, S., MCDONALD, C. & CAMPBELL, J. 2009. Little design up-front : A design science approach to integrating usability into agile requirements engineering. *Proceedings of HCI 1,* 5610**,** 549-558.

AKSU, M. & LI, C. 2010. 2010-11 World Quality Report: Trends in application quality. Available: http://www.capgemini.com/insights-and-resources/by-publication/2010-11-world-quality-report/ [Accessed 20th Oct 2011].

ANDERSON, D. J. 2010. *Kanban - Successful Evolutionary Change for Your Technology Business,* Sequim, Washington, Blue Hole Press.

BALLÉ, F. & BALLÉ, M. 2005. Lean Development. *Business Strategy Review,* 16**,** 17-22.

BARKSDALE, J. T. & MCCRICKARD, D. S. 2010. Concept mapping in agile usability: A case study. *Conference on Human Factors in Computing Systems - Proceedings***,** 4691-4694.

BECK, K., BEEDLE, M., BENNEKUM, A. V., COCKBURN, A., CUNNINGHAM, W., FOWLER, M., GRENNING, J., HIGHSMITH, J., HUNT, A., JEFFRIES, R., KERN, J., MARICK, B., MARTIN, R. C., MELLOR, S., SCHWABER, K., SUTHERLAND, J. & THOMAS, D. 2001a. *Manifesto for Agile Software Development* [Online]. Available: http://agilemanifesto.org/ [Accessed August 31st 2011].

BECK, K., BEEDLE, M., BENNEKUM, A. V., COCKBURN, A., CUNNINGHAM, W., FOWLER, M., GRENNING, J., HIGHSMITH, J., HUNT, A., JEFFRIES, R., KERN, J., MARICK, B., MARTIN, R. C., MELLOR, S., SCHWABER, K., SUTHERLAND, J. & THOMAS, D. 2001b. *Principles behind the Agile Manifesto* [Online]. Available: http://agilemanifesto.org/principles.html [Accessed August 31st 2011].

BENIGNI, G., GERVASI, O., PASSERI, F. L. & KIM, T. H. 2010. USABAGILE-Web: A web agile usability approach for web site design. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* 6017 LNCS**,** 422-431.

BERG, B. L. 2004. *Qualitative research methods for the social sciences,* Boston, Mass., Pearson.

BEYER, H., HOLTZBLATT, K. & BAKER, L. 2004. An agile customer- centered method: Rapid contextual design. *Extreme Programming and Agile Methods - XP/Agile Universe 2004***,** 527-554.

BLOMKVIST, S. 2005. Towards a Model for Bridging Agile Development and User-Centered Design

Human-Centered Software Engineering — Integrating Usability in the Software Development Lifecycle. *In:* SEFFAH, A., GULLIKSEN, J. & DESMARAIS, M. C. (eds.). Springer Netherlands.

BOEHM, B. & TURNER, R. 2003a. *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison-Wesley Professional.

BOEHM, B. & TURNER, R. 2003b. Using risk to balance agile and plan-driven methods. *Computer,* 36.

BRERETON, P., KITCHENHAM, B. A., BUDGEN, D., TURNER, M. & KHALIL, M. 2007. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software,* 80**,** 571-583.

BROOKS, F. P. 1987. No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer,* 20**,** 10-19.

BROWN, J., LINDGAARD, G. & BIDDLE, R. 2008. Stories, sketches, and lists: Developers and interaction designers interacting through artefacts. *Proceedings - Agile 2008 Conference***,** 39-50.

BUDWIG, M., JEONG, S. & KELKAR, K. 2009. When user experience met agile: A case study. *Conference on Human Factors in Computing Systems - Proceedings***,** 3075-3083.

CHAMBERLAIN, S., SHARP, H. & MAIDEN, N. 2006. Towards a Framework for Integrating Agile Development and User-Centred Design

Extreme Programming and Agile Processes in Software Engineering. *In:* ABRAHAMSSON, P., MARCHESI, M. & SUCCI, G. (eds.). Springer Berlin / Heidelberg.

CHO, L. 2009. Adopting an agile culture: A user experience team's journey. *Proceedings - 2009 Agile Conference, AGILE 2009***,** 416-421.

COLEMAN, G. & O'CONNOR, R. 2007. Using grounded theory to understand software process improvement: A study of Irish software product companies. *Information and Software Technology,* 49**,** 654-667.

CONSTANTINE, L. L. 2002. Process Agility and Software Usability: Toward Lightweight Usage-Centered Design. *Information Age*.

CONSTANTINE, L. L. & LOCKWOOD, L. A. D. 2002. Usage-centered engineering for web applications. *IEEE Software,* 19**,** 42-50.

COOPER, A. 2008. The Wisdom of Experience. Available: http://www.cooper.com/journal/agile2008/ [Accessed 12th Jan 2012].

DÉSILETS, A. 2005. Are Agile and Usability Methodologies Compatible?

DETWEILER, M. 2007. Managing UCD within agile projects. *Interactions,* 14**,** 40-42.

DYBÅ, T. & DINGSØYR, T. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology,* 50**,** 833-859.

DÜCHTING, M., ZIMMERMANN, D. & NEBE, K. 2007. Incorporating user centered requirement engineering into agile software development. *Proceedings of HCI International***,** 58-67.

FEDEROFF, M. 2008. Extreme usability: Adapting research approaches for agile development. *CHI'08: CHI'08 Extended Abstracts on Human Factors in Computing Systems***,** 2269-2272.

FEDEROFF, M. & COURAGE, C. 2009. Successful user experience in an agile enterprise environment.

FERREIRA, J., NOBLE, J. & BIDDLE, R. 2007a. Agile development iterations and UI design. *Proceedings - AGILE 2007***,** 50-58.

FERREIRA, J., NOBLE, J. & BIDDLE, R. 2007b. Up-front interaction design in agile development. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* 4536 LNCS**,** 9-16.

FERREIRA, J., SHARP, H. & ROBINSON, H. 2010. Values and assumptions shaping agile development and user experience design in practice. *Agile Processes in Software Engineering and Extreme Programming***,** 178-183.

FERREIRA, J., SHARP, H. & ROBINSON, H. 2011. User experience design and agile development: Managing cooperation through articulation work. *Software - Practice and Experience,* 41**,** 963-974.

FISHER, K. G. & BANKSTON, A. Year. From cradle to sprint: Creating a full-lifecycle request pipeline at nationwide insurance. *In*, 2009. 223-228.

FOX, D., SILLITO, J. & MAURER, F. 2008. Agile methods and user-centered design: How these two methodologies are being successfully integrated in industry. *Proceedings - Agile 2008 Conference***,** 63-72.

HAKIM, J., SPITZER, T. & ARMITAGE, J. 2003. Sprint: Agile specifications in Shockwave and Flash. *Proceedings of the 2003 conference on Designing for user experiences.* San Francisco, California: ACM.

HODGETTS, P. 2005. Experiences integrating sophisticated user experience design practices into Agile processes. *Proceedings - AGILE Confernce 2005,* 2005**,** 235-242.

HOLWEG, M. 2007. The genealogy of lean production. *Journal of Operations Management,* 25**,** 420-437.

HONIOUS, J. & CLARK, J. 2006. Something to believe in. *Agile Conference, 2006* Minneapolis, MN

HOSSEINI-KHAYAT, A., HELLMANN, T. D. & MAURER, F. 2010. Distributed and Automated Usability Testing of Low-Fidelity Prototypes. *Proceedings of the 2010 Agile Conference.* IEEE Computer Society.

HUDSON, W. 2003. Adopting User-Centered Design within an Agile Process: A Conversation. *Cutter IT Journal,* October 2003.

HUMAYOUN, S. R., DUBINSKY, Y. & CATARCI, T. 2011. A three-fold integration framework to incorporate user-centered design into agile software development. *Proceedings of the 2nd international conference on Human centered design.* Orlando, FL, USA: Springer-Verlag.

HUSSAIN, Z., LECHNER, M., MILCHRAHM, H., SHAHZAD, S., SLANY, W., UMGEHER, M. & VLK, T. 2008. User Interface Design for a Mobile Multimedia Application: An Iterative Approach. *First International Conference on Advances in Computer-Human Interaction, 2008* Sainte Luce.

HUSSAIN, Z., MILCHRAHM, H., SHAHZAD, S., SLANY, W., TSCHELIGI, M. & WOLKERSTORFER, P. 2009a. Integration of extreme programming and user-centered design: Lessons learned. *Lecture Notes in Business Information Processing,* 31 LNBIP**,** 174-179.

HUSSAIN, Z., SLANY, W. & HOLZINGER, A. 2009b. Current State of Agile User-Centered Design: A Survey

HCI and Usability for e-Inclusion. *In:* HOLZINGER, A. & MIESENBERGER, K. (eds.). Springer Berlin / Heidelberg.

HUSSAIN, Z., SLANY, W. & HOLZINGER, A. 2009c. Investigating Agile User-Centered Design in Practice: A Grounded Theory Perspective

HCI and Usability for e-Inclusion. *In:* HOLZINGER, A. & MIESENBERGER, K. (eds.). Springer Berlin / Heidelberg.

ILLMENSEE, T. & MUFF, A. 2009. 5 users every friday: A case study in applied research. *Proceedings - 2009 Agile Conference, AGILE 2009***,** 404-409.

JOKELA, T. & ABRAHAMSSON, P. 2004. Usability Assessment of an Extreme Programming Project: Close Co-operation with the Customer Does Not Equal to Good Usability

Product Focused Software Process Improvement. *In:* BOMARIUS, F. & IIDA, H. (eds.). Springer Berlin / Heidelberg.

KNIBERG, H. & SKARIN, M. 2010. Kanban and Scrum. *making the most of both.* C4Media Inc.

KOLLMANN, J., SHARP, H. & BLANDFORD, A. 2009. The importance of identity and vision to user experience designers on agile projects. *Proceedings - 2009 Agile Conference, AGILE 2009***,** 11-18.

LARMAN, C. & BASILI, V. R. 2003. Iterative and Incremental Development: A Brief History. *Computer,* 36**,** 47-56.

LEE, J. C., JUDGE, T. K. & MCCRICKARD, D. S. 2011. Evaluating eXtreme scenario-based design in a distributed agile team. *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems.* Vancouver, BC, Canada: ACM.

LEE, J. C., MCCRICKARD, D. S. & STEVENS, K. T. 2009. Examining the Foundations of Agile Usability with eXtreme Scenario-Based Design. *Proceedings of the 2009 Agile Conference.* IEEE Computer Society.

LESZEK, A. & COURAGE, C. 2008. The Doctor is "In" -- Using the Office Hours Concept to Make Limited Resources Most Effective. *Proceedings of the Agile 2008.* IEEE Computer Society.

MECKEM, S. & CARLSON, J. L. 2010. Using "rapid experimentation" to inform customer service experience design. *Conference on Human Factors in Computing Systems - Proceedings***,** 4553-4565.

MEMMEL, T., GUNDELSWEILER, F. & REITERER, H. 2007. Agile human-centered software engineering. *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it - Volume 1.* University of Lancaster, United Kingdom: British Computer Society.

MESZAROS, G. & ASTON, J. 2006. Adding usability testing to an agile project. *Proceedings - AGILE Conference, 2006,* 2006**,** 289-294.

MILLER, L. 2005. Case study of customer input for a successful product. *Proceedings - AGILE Confernce 2005,* 2005**,** 225-234.

MILLER, L. & SY, D. 2009. Agile user experience SIG. *Conference on Human Factors in Computing Systems - Proceedings***,** 2751-2754.

MOE, N. B., DINGSØYR, T. & DYBÅ, T. 2008. Understanding Self-Organizing Teams in Agile Software Development. *Proceedings of the 19th Australian Conference on Software Engineering.* IEEE Computer Society.

NAJAFI, M. & TOYOSHIBA, L. 2008. Two case studies of user experience design and agile development. *Proceedings - Agile 2008 Conference***,** 531-536.

NIELSEN NORMAN GROUP. 2010. *User Experience - Our Definition* [Online]. Available: http://www.nngroup.com/about/userexperience.html [Accessed 20th Jan 2011].

NODDER, C. & NIELSEN, J. 2008. Agile usability: best practices for user experience on agile development projects. Available: http://www.nngroup.com/reports/agile.

NORMAN, D., MILLER, J. & HENDERSON, A. 1995. What you see, some of what's in the future, and how we go about doing it: HI at Apple Computer. *Conference companion on Human factors in computing systems.* Denver, Colorado, United States: ACM.

OATES, B. J. 2006. *Researching information systems and computing,* London, Sage Publications.

OBENDORF, H. & FINCK, M. 2008. Scenario-based usability engineering techniques in agile development processes. *Conference on Human Factors in Computing Systems - Proceedings*, 2159-2166.

PATTON, J. 2005. Finding the forest in the trees. *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications.* San Diego, CA, USA: ACM.

PATTON, J. 2008a. Getting software RITE. *IEEE Software,* 25**,** 20-21.

PATTON, J. 2008b. Twelve emerging best practices for adding UX work to Agile development. *AgileProductDesign.com* [Online]. Available from: http://www.agileproductdesign.com/blog/emerging_best_agile_ux_practice.html [Accessed 27th June 2008 2011].

PATTON, J. 2009. Kanban Development Oversimplified. Available: http://agileproductdesign.com/blog/2009/kanban_over_simplified.html [Accessed 20th Jan 2012].

POLLICE, G. 2006. Does process matter? *The Rational Edge* [Online]. Available: http://www.ibm.com/developerworks/rational/library/aug06/pollice/index.html [Accessed May 28th 2011].

POPPENDIECK, M. & POPPENDIECK, T. 2003. *Lean Software Development - An Agile Toolkit*, Addison-Wesley.

RAMSAY, A. 2010. *Findings from the State of Agile UX Survey* [Online]. Available: http://www.andersramsay.com/2010/11/06/findings-from-the-state-of-agile-ux-survey [Accessed Jan 20th 2011].

SALAH, D. Year. A framework for the integration of user centered design and agile software development processes. *In:* 33rd International Conference on Software Engineering, 2011 Waikiki, Honolulu, Hawaii. 1132-1133.

SCHWABER, K. 1995. SCRUM Development Process. *OOPSLA'95 Workshop on Business Object Design and Implementation* [Online]. Available: http://home.hib.no/ai/data/master/mod251/2009/articles/scrum.pdf [Accessed 20th Jan 2011].

SCHWABER, K. 2004. *Agile project management with Scrum,* Redmond, Wash., Microsoft Press.

SCHWABER, K. & BEEDLE, M. 2002. *Agile software development with Scrum,* Upper Saddle River, NJ, Prentice Hall.

SCHWABER, K. & SUTHERLAND, J. 2011. The Scrum Guide. Available: http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf.

SHENTON, A. K. 2004. Strategies for ensuring trustworthiness in qualitative research projects. *Education for Information,* 22**,** 63-75.

SILVA, T. S. D., MARTIN, A., MAURER, F. & SILVEIRA, M. 2011. User-Centered Design and Agile Methods: A Systematic Review. *AGILE Conference (AGILE), 2011.* Salt Lake City, UT, USA.

SOHAIB, O. & KHAN, K. Year. Integrating usability engineering and agile software development: A literature review. *In*, 2010. V232-V238.

SY, D. 2007. Adapting Usability Investigations for Agile User-centered Design. *Journal of Usability Studies,* 2**,** 112-132.

TURNER, R. & BOEHM, B. 2003. People Factors in Software Management: Lessons From Comparing Agile and Plan-Driven Methods. *CrossTalk - the Journal of Defense Software*.

UNGAR, J. 2008. The design studio: Interface design for agile teams. *AGILE Conference Agile 2008***,** 519-524.

UNGAR, J. M. & WHITE, J. A. 2008. Agile user centered design: Enter the design studio - A case study. *Conference on Human Factors in Computing Systems - Proceedings*, 2167-2177.

WEST, D. & GRANT, T. 2010. Agile Development: Mainstream Adoption Has Changed Agility. Forrester Research, Inc.

WILLIAMS, H. & FERGUSON, A. 2007. The UCD perspective: Before and after agile. *Proceedings - AGILE 2007*, 285-290.

WINTER, J., R, K., AHLBERG, R. & HOTCHKISS, J. 2011. Meeting organisational needs and quality assurance through balancing agile and formal usability testing results. *Proceedings of the Third IFIP TC 2 Central and East European conference on Software engineering techniques.* Brno, Czech Republic: Springer-Verlag.

YIN, R. K. 2003. *Case study research : design and methods* (3rd ed.), Thousand Oaks, Calif., Sage.

YIN, R. K. 2009. *Case Study Research : Design and Methods* (4th ed.), SAGE Publications, Inc.

APPENDIX A: THE AGILE MANIFESTO

The Agile manifesto was created in 2001 during a meeting of seventeen software professionals representing several different Agile flavors. The manifesto reads as follows[29]:

"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."

Additionally, twelve principles behind the manifesto are listed in the following order (the numbering is added by the author for ease of reference)[30]:

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.

[29]The original web site for the manifesto can still be reached at www.agilemanifesto.org [last accessed Oct. 30, 2011]

[30]Source: http://agilemanifesto.org/principles.html [last accessed Oct. 30, 2011]

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.

h

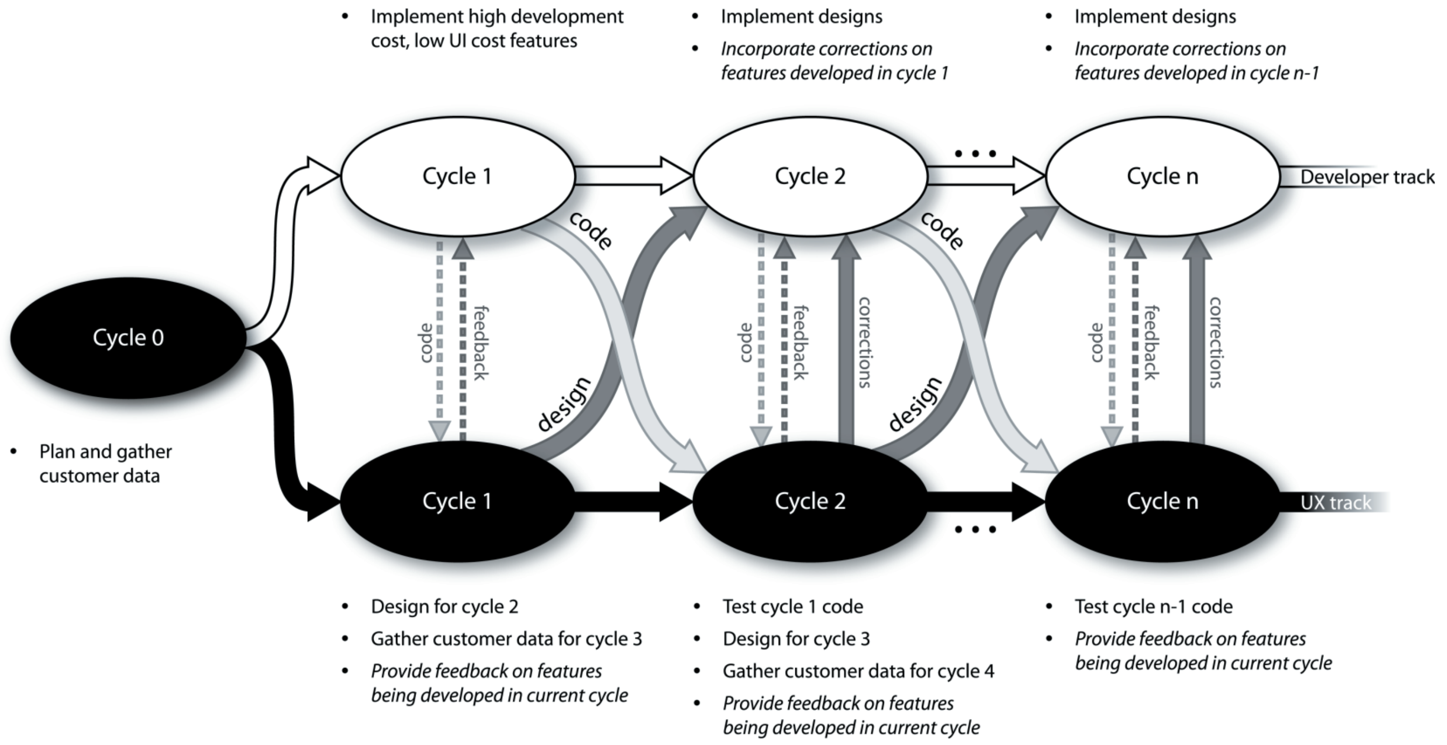
APPENDIX B: COCKBURN LEVELS

Cockburn's three levels of software method understanding, as revised by Boehm and Turner (2003a)

One of the key factors that should be considered when choosing a process to fit a given project setting, is the level of skill and understanding required by the people involved in order to use, tailor, adapt or revise a method. In this context, Alistair Cockburn has identified three levels of software method (process) understanding that help sort out what various levels of people can be expected to do within a given method framework. The levels have been slightly modified by Boehm and Turner (2003a, 2003b) in order to address some distinctions between agile and plan-driven processes (level 1 is split into 1A and 1B, and the level -1 has been added).

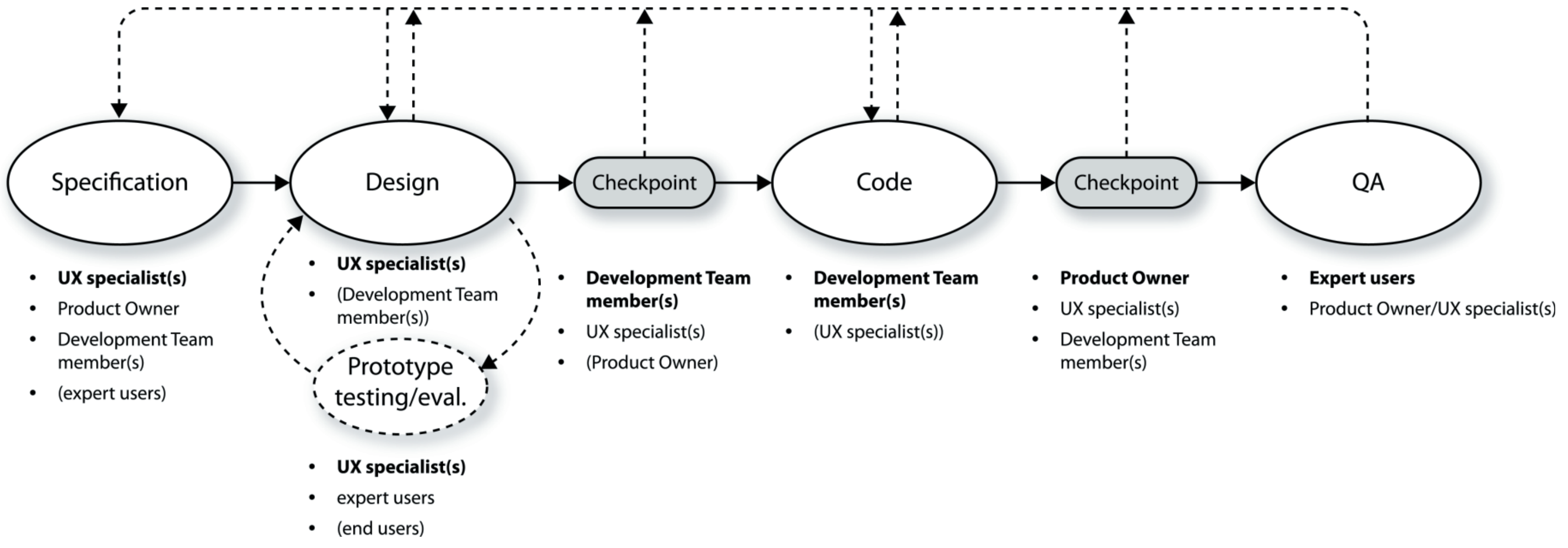
Cockburn Level	**Characteristics**
3	Able to revise a method (break its rules) to fit an unprecedented new situation
2	Able to tailor a method to fit a precedented new situation
1A	With training, able to perform discretionary method steps (e.g., sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With experience, can become Level 2.
1B	With training, able to perform procedural method steps (e.g., coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience, can master some Level 1A skills.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.

APPENDIX C: THE PARALLEL TRACK MODEL


• Implement high development cost, low UI cost features
• Implement designs
• *Incorporate corrections on features developed in cycle 1*
• Implement designs
• *Incorporate corrections on features developed in cycle n-1*
Cycle 1
Cycle 2
Cycle n
Developer track
Cycle 0
code
feedback
design
corrections
• Plan and gather customer data
Cycle 1
Cycle 2
Cycle n
UX track
• Design for cycle 2
• Gather customer data for cycle 3
• *Provide feedback on features being developed in current cycle*
• Test cycle 1 code
• Design for cycle 3
• Gather customer data for cycle 4
• *Provide feedback on features being developed in current cycle*
• Test cycle n-1 code
• *Provide feedback on features being developed in current cycle*

This version of the parallel track model is an adapted combination of the versions found in Sy (2007) and Silva et al. (2011). The main difference between the two is that in the latter, the continuous communication between designers and developers has been made explicit in the model (arrows added), and it also includes the "corrections" element (developers correcting features developed in the previous cycle based on feedback from the designers in the current cycle). These elements are included in the present version. The version by Silva et al. also includes lists of artifacts/techniques used, but these have been omitted in the present version for the sake of simplicity/clarity. Explanations set in italics are adapted from the Silva et al. version, the rest are from the Sy version.

APPENDIX D: THE LIFE CYCLE OF A USER STORY, DESCRIPTIVE MODEL



Although time frames vary, a user story usually passes through at least four general phases and two checkpoints. In the specification phase, the user story and acceptance criteria are written. In the design phase (often seen as part of the specification phase), UI designs and prototypes are created, and in some cases the designs are tested and iterated based on feedback from testing/evaluation of the prototypes. In all of the cases, the stories and designs are validated in some way by the developers before development begins, and the finished code is validated by the UX specialist(s) and/or PO before the story is passed on to QA by expert users. These checkpoints may be formal or informal: the second checkpoint is usually an explicit part of the process, while the first is informal, but in cases A and D the first seems to have been formalized as well.

For each phase and checkpoint, the role or group that is mainly responsible for creating the related work product and/or making the decision to move the story forward or backward in the process is set in bold type. Roles/groups that are usually involved are set in regular type, while roles/groups that are not always involved (or only sporadically) are shown in parentheses.

Detailed Table Of Contents