Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Isa Ojanen

Introduction and integration of UCD and Scrum methodologies

Tools and techniques

Master's Thesis
Espoo, March 27., 2016

Supervisor: Professor Marko Nieminen
Advisor: Petri Mannonen M.Sc. (Tech.)

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Isa Ojanen

Title:
Introduction and integration of UCD and Scrum methodologies Tools and techniques

Date: March 27., 2016 **Pages:** 94

Major: Usability and User Interfaces **Code:** T-110

Supervisor: Professor Marko Nieminen

Advisor: Petri Mannonen M.Sc. (Tech.)

Though the integration of user-centered development (UCD) and software development is a reasonably well-research field, there seems to be a lack of papers focusing on companies that have no UCD expert or team. In this paper we detail our research with three teams in one such company, and answer the research questions "how can UCD be introduced and integrated into Scrum development in a company with very few or no UX professionals" and "what kind of UCD and UX tools would developers find valuable when developing software".

The state of user-centered development in two of the teams, the Voyage Optimization (VO) team and the Safety team, was researched by interviewing developers and product owners, after which the recorded interviews were roughly transcribed and common themes were searched for and recognized. Two workshops were organized with the Safety team and one workshop with the VO team. As the third team, Hull Finland, had already begun experimenting with UCD with the help of hired consultants, two of their developers were interviewed in regards to their experiences with the integration process. The recording of these interviews failed and only notes taken during the interview were used in this research.

All three workshops held were unique, but shared the same outline: explanation of what UCD is and why it could provide useful in software development, introduction of selected tools and some experimentation with them, and, in two of the workshops, a developer from the Hull team joined to tell of his experiences with UCD in their team's integration process.

The answer to the research questions consists of five recommendations to the introduction and integration process: 1. Employment of a seasoned UX professional, who is also well-versed in change management and leadership 2. writing UX methods and milestones into the process description, 3. Creation of Personas, 4. Improved communication across teams regarding UX, and 5. Establishing a permanent UX team.

Keywords: User-centered development, User experience, Agile, Scrum

Language: English

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Isa Ojanen		
Työn nimi:	Käyttäjäkeskeisen tuotekehityksen perehdytys sekä integrointi Scrum-metodologiaan Työkalut ja tekniikat		
Päiväys:	27. maaliskuuta 2016	**Sivumäärä:**	94
Pääaine:	Käyttöliittymät ja käytettävyys	**Koodi:**	T-110
Valvoja:	Professori Marko Nieminen		
Ohjaaja:	Diplomi-insinööri Petri Mannonen		

Vaikka käyttäjäkeskeinen tuotekehitys (user-centered development, UCD) sekä ohjelmistokehitys ovat molemmat suhteellisen hyvin tutkittuja tietotekniikan osa-alueita, niin vain harvat tutkimukset keskittyvät yrityksiin, joissa ei ole omaa käyttäjäkokemukseen (user experience, UX) keskittyvää työryhmää tai UCD:n asiantuntijaa. Tässä tutkimuksessa työskentelemme kolmen työryhmän kanssa yhdessä tällaisessa yrityksessä. Pyrimme vastaamaan tutkimuskysymyksiin ”miten käyttäjäkeskeinen tuotekehitys voidaan tuoda ja integroida Scrum-tuotekehitykseen”sekä ”millaiset käyttäjäkeskeisen tuotekehityksen sekä käyttökokemukseen keskittyvät työkalut voisivat olla hyödyksi ohjelmistokehittäjille”.

UCD:n taso kahdessa eri työryhmässä kartoitettiin haastattelemalla työryhmien jäseniä ja tunnistamalla haastatteluista työryhmittäin toistuvat teemat. Yhteistyötä tehtiin myös kolmannen työryhmän kanssa, jossa UCD:n integroiminen Scrum-ohjelmistokehitysprosessiin oli jo aloitettu ulkopuolisten konsulttien avustuksella. Työryhmän kahta jäsentä haastateltiin heidän hyödyllisiksi kokemistaan työkaluista sekä hyviksi koettujen käytäntöjen suhteen.

Tutkimuksessa järjestettiin kolme työpajaa, jotka perustuivat samalle kaavalle: ensin käytiin läpi mitä UCD on ja mitä hyötyä siitä voi olla ohjelmistokehityksessä, jonka jälkeen esiteltiin valitut työkalut sekä tehtiin joitakin käytännön harjoituksia niiden kanssa. Toinen haastateltu Hull-työryhmän jäsen otti osaa myös kahteen työpajaan, ja kertoi omista kokemuksistaan UCD:n kanssa.

Haastattelujen ja työpajojen pohjalta vastauksena tutkimuskysymyksiin suosittelemme: 1. sellaisen kokeneen UX-ammattilaisen palkkaamista, joka on myös harjaantunut muutosjohtaja, 2. UX metodien ja virstapylväiden kirjoittamista osaksi prosessikuvausta, 3. Persoonien luomista, 4. kommunikaation parantamista UX:n suhteen työryhmien välillä, sekä 5. pysyvän UX-työryhmän luominen.

Asiasanat:	Käyttäjäkeskeinen tuotekehitys, Käyttökokemus, Agile, Scrum
Kieli:	Englanti

Acknowledgements

First and foremost, I wish to thank my parents, who have supported me through my whole life and encouraged me to find my own field of study through trial and error. I also thank my friends, providing good ways to avoid studying and making my years at the university unforgettable. My instructor Petri Mannonen was admirably patient with my work, providing help consistently along the way, and I thank also my professor Marko Nieminen.

This thesis would not have been possible without Mikko Kuosa from Napa, and I deeply thank him and the teams I worked with for allowing me to complete my studies with them.

Finally, I thank my fiancé for always being there.

Espoo, March 27., 2016

Isa Ojanen

Abbreviations and Acronyms

DoD	Definition Of Done
PM	Product Manager
PO	Product Owner
UCD	User-Centered Development
UI	User Interface
UX	User Experience
VO	Voyage Optimization

Contents

Chapter 1

Introduction

In this chapter we go over the motivation for the research and the research questions. We then provide a summary of the findings and finally present an overview of the structure of the thesis.

1.1 Problem motivation

Combining user-centered development (UCD) and software development methods is a reasonably well researched field, with many papers from the 1990s, e.g.: [Katz-Haas, 1998; Mayhew, 1999; Heinbokel et al., 1996]

As the trend in software development processes has become more agile-focused, there has likewise been an increase in papers researching agile methods and UCD. These papers look at methodologies such as XP and Scrum, and most often how a company with a user-experience (UX) team can integrate UCD into their agile software development process in a more efficient way. [Sy, 2007; Budwig et al., 2009; Fox et al., 2008]

Combining UCD and the traditional waterfall software development process is vastly different from combining it with the iterative and incremental agile development. The waterfall method consists of distinct phases, as seen in 1.1, with feedback loops back into a previous phase. The development process is heavily document-driven, relying on completed documentation early in the development process as discovered by Boehm [1988]. In the context of their research paper, using the waterfall method would mean that UX and usability requirements would be defined well before development begins, and in a perfect situation any decisions would not be revised after the decisions had been made.

In practice, then, integrating UCD into a waterfall process would mean heavy, up-front design. If usability testing is done, it would be completed

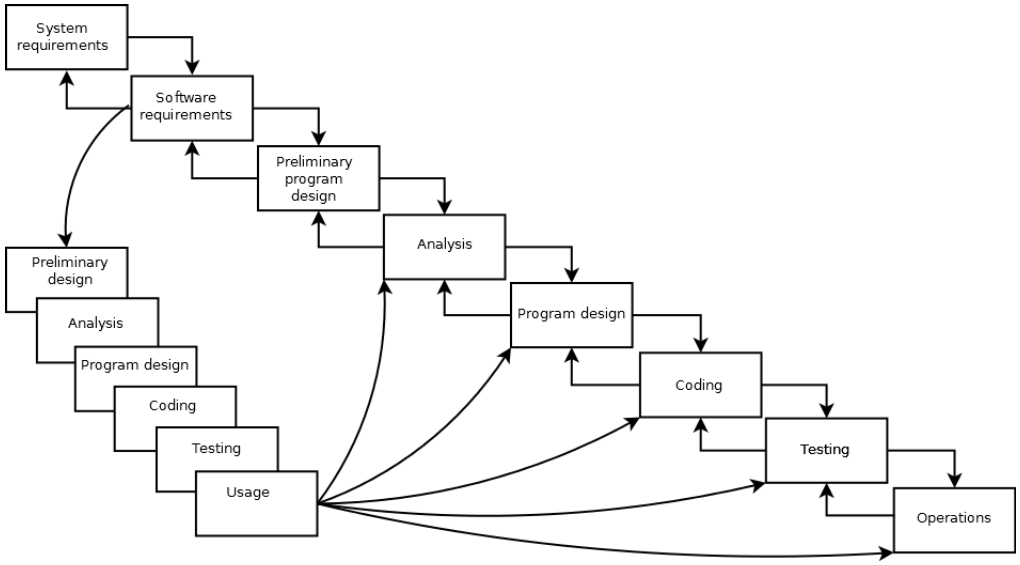

Figure 1.1: Waterfall Development Process as created by Winston Royce (1970) [Boehm, 2006]

before implementation begins, with wireframes or other tools, depending on the needs, schedule, time constraints, budget, and so on, of the project.

The Agile manifesto, on the other hand, values "Working software over comprehensive documentation" and defines one of the basic principles of agile software development as "Welcome changing requirements, even late in development". Additionally, agile processes "harness change for the customer's competitive advantage". [Beck et al., 2001b] Though a web page, the trustworthiness and security of the site is good, as the Agile Alliance is a global nonprofit organization. As it is committed to advancing Agile development, we trust the information on their website to have value and correct information.

One agile software development process is Scrum development, which is iterative in that features and items to be done are chosen from the backlog every 2-4 weeks, depending on the team, and up-front documentation and design is avoided. The whole product is thus done in small increments. So, in Scrum, UCD requirements could be researched just before the implementation cycle begins, so that the requirements are up-to-date as design and development begins. This would also mean that the requirements research and UX design is not done all at once, but rather in increments as the iterative development moves on.

While constructing the literature review of this research we came across multiple papers that described how UCD and agile software can be integrated

in teams, which have at least one UCD expert (e.g. [Sy, 2007; Fox et al., 2008; Ungar and White, 2008]), but only Fox et al. [2008] mentioned a team that had no UCD specialists. Some other papers did not determine if the methods and tools should be used by UCD experts or if the integration process should include UCD experts (e.g. [Holtzblatt et al., 2004; Singh, 2008]). Due to the difficulties in finding papers focusing on teams and companies with very small or non-existent UCD teams, we've come to the conclusion that more research in this area is needed.

This research focuses on one such software development company who were in the process of combining UCD into their Scrum development process, while having multiple software development teams but only one UX-focused employee.

1.2 Research environment

The research took place at Napa, a software company that creates products for ship planning, building and steering. At the time the research began the Napa had one, just-hired UX specialist and a small team of outsourced UX consultants working part-time with the "Hull" development team. In total, this research focuses on three teams and cases:

1. The Safety team: Introducing UCD gradually over a long time period
2. The Voyage Optimization (VO) team: Introducing UCD in one workshop session
3. The Hull Finland team: Outsourcing UX knowledge and work to a consultancy

The software development process at Napa follows the Scrum methodology. Each team has a Product Owner (PO) and a Product Manager (PM), of whom the PO works more closely with the team and manages the backlog. The outsourced consultants work mainly with the Hull team, providing input and help in transforming their development process to better facilitate UX activities and design, but also provide sketches, prototypes and wireframes. The consultants have also worked with other teams at Napa, but these have been on a more case-by-case basis.

	Inspection Methods			Test Methods		
	Heuristic Evaluation	Cognitive Walkthrough	Action Analysis	Thinking Aloud	Field Observation	Questionnaires
Applicably in Phase	All	All	Design	Design	Final Testing	All
Required Time	Low	Medium	High	High	Medium	Low
Needed Users	None	None	None	3+	20+	30+
Required Evaluators	3+	3+	1 - 2	1	1+	1
Required Equipment	Low	Low	Low	High	Medium	Low
Required Expertise	Medium	High	High	Medium	High	Low
Intrusive	No	No	No	Yes	Yes	No

Comparison of Usability Evaluation Techniques

Figure 1.2: Comparison of usability evaluation techniques [Holzinger, 2005]

1.3 Research questions

As written above, there seems to be little research on how UCD and agile software development can be integrated in companies that have very small or no UX team or personnel at all. Thus, the first research question seek an ansswer to is "how can UCD be introduced and integrated into Scrum development in a company with very few or no UX professionals". Two of the three teams we collaborated with did not have regular UX input from a UX professional, and UCD and UX methods were not well-known in either team.

In addition to this, we aim to answer the question "what kind of UCD and UX tools would developers find valuable when developing software".

The research of Holzinger [2005] evaluated the level of expertise, among other attributes, required by various usability evaluation techniques. As seen from figure 1.2 only questionnaires are considered to require low expertise. In a development team with no UX professionals other evaluation methods could thus prove to be challenging to apply. Vaananen-Vainio-Mattila et al. [2008] analyzed further that there seems to be a gap between the academic and industrial interest in UX, as seen in figure 1.3

As listed in chapter two, there are multiple tools and methods that can be used to improve the user experience and usability of a product. Our second goal in this research is to evaluate their applicability to a team without a UX professional.

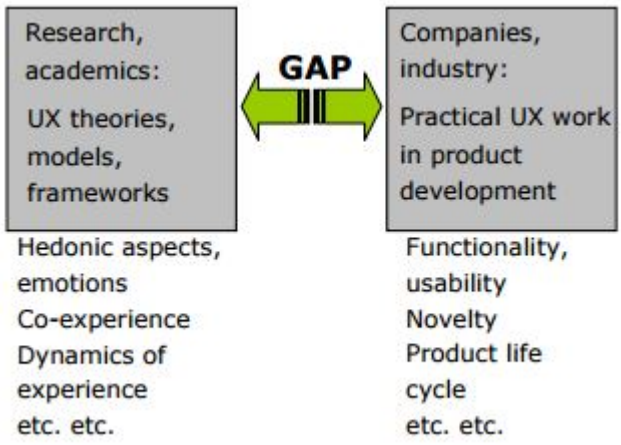

Figure 1.3: Vaananen-Vainio-Mattila et al. [2008] analyzed how UX research and industrial UX development currently focus on different issues

Chapter 2

UCD in Agile Software development

In this chapter we will go through what different components user-centerd development (UCD) and agile software development entail, and what various aspects should be considered when integrating the two frameworks. Later we go through what kind of methods, methodologies and tools have been used both in business and academia to achieve integration.

2.1 UCD, Usability, UX, and Agile Software Development

To understand how user-centered design (UCD) and agile software development methods can be integrated, one must understand what they consists of. In this section we go over what UCD, Usability, UX and Agile software development are.

2.1.1 User-Centered Development

Abras et al. [2004] define UCD as a term for a broad combination of philosophy and methods, ultimately describing design processes in which the end-user has a central position. They determine further that it consists of methods of both extremes: methods, that involve the user in the whole development process, from concept to release, and methods, that involve the users only at certain times, such as during the capturing and elicitation of requirements. This view of UCD being a broad methodology is shared by Mao et al. [2005], who define UCD as a "multidiciplinary design approach

based on the active involvement of users to improve the understanding of user and task requirements, and the iteration of design and evaluation".

The importance of UCD to software development is further verified by its including in the international ISO standard [Mao et al., 2005].

2.1.2 Usability and User Experience

Brooke [1996] sums usability as "being a general quality of appropriateness to a purpose of any particular artefact", meaning, that usability of any product should be viewed and evaluated in the context of its meant use. However, this is by far not the only definition of usability, and the actual abundance of definitions poses a problem, as it makes the concept of usability confusing [Seffah and Metzker, 2004]. In their paper Seffah and Metzker [2004] describe three definitions of usability:

- "The capability of the software product to be understood, learned, used, and attractive to the user, when used under specified conditions." (ISO 9126. Software Product Evaluation: Quality Characteristics and Guidelines for their Use)
- "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use." (ISO/DIS 9241-11. Guidance on Usability. Ergonomic Requirements for Office Work with Visual Display Terminals (VDT). 1996.)
- "The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component." (IEEE Std. 1061. Software Quality Metrics Methodology, 1998)

In addition to these, usability is defined in, e.g., the standard ISO 9241, which describes usability as the "extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [International Organization for Standardization, 2010], and by multiple other researchers and professionals. Jokela, for example, states that one of the best known definitions is by Nielsen, and goes on to describe it as "usability is about learnability, efficiency, memorability, errors and satisfaction" [Jokela et al., 2003].

The same difficulties can also be found when trying to define user experience (UX). At UX Definitions there are over 25 definitions for UX, ranging

from Sutcliffe's "Users' judgement of product quality arising from their experience of interaction, and the product qualities which engender effective use and pleasure" to Nielsen-Norman Group's

> "All aspects of the end-user's interaction with the company, its services, and its products. – 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". [UX Definitions]

Though a web page, [UX Definitions] is upheld by UX professionals and we consider it to be trustworthy. As not all of the source links in the 25 definitions work, the page should still be considered with some precaution.

Furthermore, in ISO 9241-210 user experience is defined as "a person's perceptions and responses that result from the use and/or anticipated use of a product, system or service."

One possible reason why the interest in UX has grown is the fact that deficiencies of traditional usability framework - its focus mainly on user cognition and user performance - have become more known in the human-computer interaction (HCI) community. In contrast to the focus areas of usability, UX shifts the focus to "user affect, sensation and the meaning as well as value of such interactions in everyday life". [Law et al., 2009]

Bevan [2009] writes that the difference of UX and usability is due to the different emphasis between task performance and pleasure, leading to different concerns during development. He goes on to define four typical concerns in the context of usability and two in context of UX, but clarifies that while some organizations keep the two sets of issues separate and under different headings, some organisations group both sets under the shared umbrella of user experience. Quotations of the different concerns of usability and UX have been summarized in table 2.1.

Concerns with usability	**Concerns with UX**
"Designing for and evaluating user comfort, overall effectiveness, efficiency, and satisfaction."	"Understanding and designing the user's experience with a product: the way in which people interact with a product over time: what they do and why."

"Designing to make the product easy to use, and evaluating of it to identify and fix usability problems."	"Maximising the achievement of the hedonic goals of stimulation, identification and evocation and associated emotional responses"
"When relevant, the temporal aspect leads to a concern for learnability."	

Table 2.1: Different concerns of usability and UX as described by Bevan [2009]

In their research Law et al. [2009] asked 275 UX researchers and practitioners their views on UX. The results indicated that usability is viewed as an essential precondition to good UX by 269 persons. Additionally, the respondents mostly agreed that UX is a dynamic, content-dependent and subjective concept, based on the multiple potential benefits a user could gain from a product. It is viewed as a new concept, founded from (grounded in) UCD practices, and a part of the HCI field.

2.1.3 Agile Software Development

Agile Software Development is a software process methodology which key values are "individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation and responding to change over following a plan". It includes many principles from lean manufacturing, and was developed as an answer to the failures of then-popular software development project management paradigms, such as the waterfall-process. [Beck et al., 2001a]

Created in 2001 by Beck et al. [2001b], the agile development methodology is founded on 12 principles, and could be summarized as iterative and incremental development process that welcomes changes and demands business and user-input. However, these twelve steps and the key values do not provide any concrete steps on how to do software development in an agile way. Two of the methodologies that share the values of agile and do offer actual steps to follow are eXtreme Programming (XP) and Scrum [Beck et al., 2001a]. The evolution of the waterfall-method into XP can be seen in 2.1.

In chapter 2.4, where we describe research done regarding agile methods and UCD integration, these two methodologies are the most prominent software development processes, and so we will briefly go over their principles here.

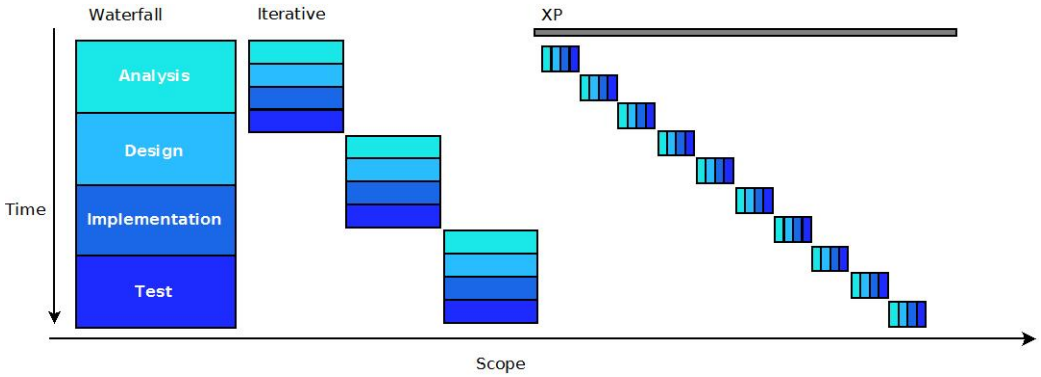

Figure 2.1: From waterfall to XP, Beck [1999]

eXtreme Programming

eXtreme Programming is the most widely used agile process [Beck et al., 2001a], and it consists of 13 major practices [Beck, 1999].

The XP development cycle can bee seen in 2.2:

The process begins with the customer picking the most valuable features, called Stories in XP, to be developed in the new release. The factors the customer weighs when choosing the stories are the costs of the stories and how fast the team implements stories. The customer then chooses the next iteration's stories from the stories still remaining in the release, again basing their choice on the costs and the team's speed. It is then the programmers turn to split the stories into smaller tasks, which are divided among the developers. Each programmer then creates a set of test cases, based on a task, that prove that a task is finished. Finally, the programmer works with a partner to run the test cases, and develops the design, while keeping the solution as simple as possible for the system as a whole. [Beck, 1999]

Applicability of XP, as described by Beck [1999], are "outsourced or in-house development of small- to medium-sized systems where requirements are vague and likely to change. " [Beck, 1999]

Scrum

Scrum, as defined in the official Scrum Guide, is "A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value." [Scrum Guide]. It was developed in the early 1990s by Jeff Sutherland and Ken Schwaber, who later took part in the creation of The Agile Manifesto, and nowadays Scrum is considered an agile framework [Scrum Guide].

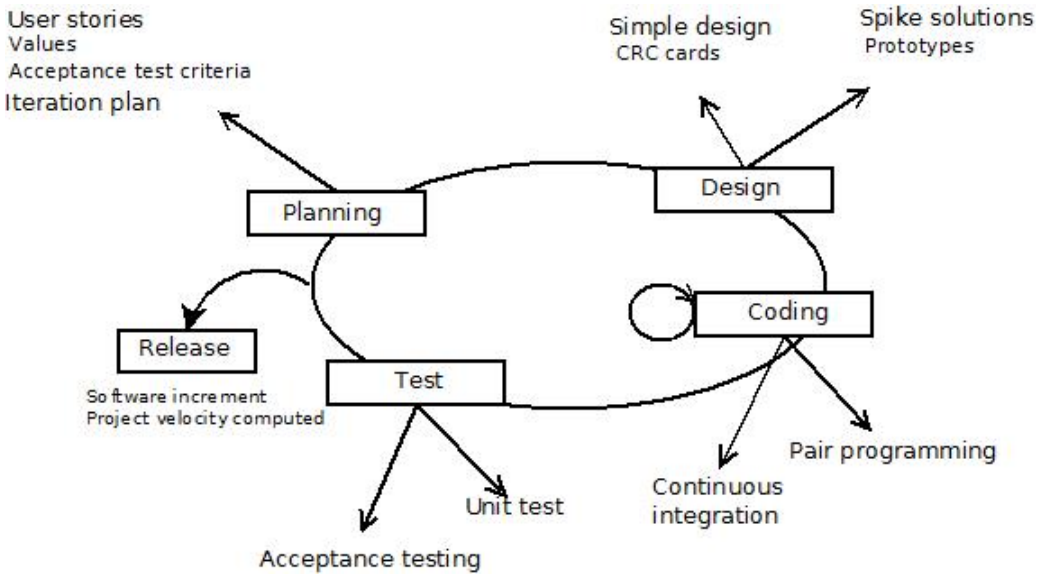

Figure 2.2: XP Development Process, Beck et al. [2001a]

The overall process and artefacts of Scrum are, shortly put, as follows:

The Product Backlog holds all features, functions, requirements, enhancements and fixes that might be needed in the product, in order of importance. The Product Backlog is maintained by th Product Owner, who is responsible for the list's content, availability and ordering. The Backlog is never complete, as it evolves with the project and product development. The Product Owner and the Development Team are both responsible for refining the Backlog by reviewing and revising the items in it.

From the Product Backlog items are chosen to the Sprint Backlog, which also includes a plan on how the product Increment will be delivered and the Sprint Goals met. In essence, the Sprint Backlog contains the items that the Development Team hopes to complete in the Sprint. The Backlog can be modified only by the Development Team during the Sprint, by, for example, adding new items as needed to the Backlog.

The Sprint is a time-boxed event that lasts at most a month, and usually another Sprint begins immediately afterwards. The work to be done in the Sprint is decided in the planned at a Sprint Planning, an eight hour long session at the longest, and held at the beginning of the Sprint. Each Sprint includes a definition of what will be done, and can be considered as a month-long project.

During the Sprint Daily Scrums, time-boxed meetings of maximum 15 minutes in length, are held. The whole Development Team takes part in the

meetings to synchronize activities and to create a plan for the next 24 hours. The plan is created by inspecting the work done after the previous Daily Scrum and estimating what could be done before the next one.

As a Sprint nears its end, a Sprint Review is held to evaluate the developed Increment and to make adaptations to the Product Backlog as needed. In this informal meeting the Stakeholders and the Scrum Team discuss what was done in the Sprint and what could be done in the future to optimize value. The meeting is time-boxed to four hours maximum.

Finally, also at the end of a Sprint, the Scrum Team holds a Sprint Review, where they have the opportunity to inspect the team itself and create a plan how the team and its work could be improved in the next Sprint. [Scrum Guide]

Though the information is from a website, the site claims to contain the official Scrum Guide both as downloadable and online versions. Even if websites as such should not be considered 100% trustworthy,a as we see no evidence of a hoax and we have come across similar information before, we consider the website to be relatively trustworthy and to have correct information.

2.2 Introducing UCD to Agile

In this section we go over why it is beneficial to combine UCD with agile software development, what challenges there inherently are in combining the two methodologies, and, finally, what similarities the two methodologies share.

2.2.1 The need for UCD in agile software development

While the need to know and understand users has widely been accepted and recognized in software development, there is dispute in the intensity of the users' involvement in the development process - what role a user should play, and how and how much they should be involved [Chamberlain et al., 2006]. As usability is becoming an increasingly more important sales criteria for products [Düchting et al., 2007], organizations and development processes must adapt and make decisions to answer the need. As Vaananen-Vainio-Mattila et al. [2008] write, "User-centered development (UCD) is still the key to designing for good user experiences. We must understand users' needs and values first, before designing and evaluating solutions".

Usability Engineering activities in XP and Scrum were investigated by Düchting et al. [2007] through a gap-analysis, and their work was based on the Requirement Engineering framework of Zimmermann & Grötzbach

(as cited in [Düchting et al., 2007]). The framework generates three types of requirements, Usability Requirements, Workflow Requirements, and User Interface Requirements. Based on this framework, Duchting et al analyzed if and how Scrum and XP methodologies consider the different requirements [Düchting et al., 2007].

The outcome of the research was that both agile models were found lacking in regards to handling User-Centered Requirements, with most insufficiencies in the Usability Requirements [Düchting et al., 2007].

Another challenge that agile methods can be considered to have is the role of a customer representative [McInerney and Maurer, 2005], who might not be an actual user. As Sy [2007] explains, a customer "is a role filled by one or more members of the product team", and the responsibilities of this role include representing the end-user. As one of the Usability Requirements neither XP nor Scrum fulfilled is observation of the user in the context of use [Düchting et al., 2007], serious issues could be caused by not knowing the user's true needs.

Additionally, McInerney and Maurer [2005] describe how agile literature regards understanding users in the software development process as such: Agile methods favour developers working with a customer representatives, who in turn make final decisions to the system based on the developers' recommendations. Only a preliminary research is completed before beginning development, and customer needs are clarified as questions rise during development.[McInerney and Maurer, 2005]

Finally, Sohaib and Khan [2010] sum the challenge agile methods face regarding user-knowledge: "The major challenge for an agile approach is how to identify the requirements of a system as accurately as possible from a customer who is not the actual end user." Lack of integration between agile development and UCD can thus result in e.g. the organization not understanding to take both the expert and novice users' needs into account [Sohaib and Khan, 2010].

One solution to the issues mentioned above was discovered by Nielsen [1994], who concluded from his research that even a small-scale empirical study can be of much value and assistance to non-human factors people as they evaluate user interfaces.

2.2.2 The difficulties of combining UCD and agile

One of the challenges of introducing usability and UCD to agile software development is the fact that the term "usability" has been given multiple interpretations and definitions over the years [Seffah and Metzker, 2004]. When there is no one single term to explain, introducing a completely new

framework becomes even more difficult.

A second challenge is the difference in the amount of suggested up-front work - Chamberlain et al. [2006] determine that whereas user-centred development (UCD) encourages knowing the user as much beforehand as possible, agile methods are generally against heavy up-front work. They repeat this understanding in reporting the difference of preferred amount of documentation, as UCD advocates request certain design products while agile methods prefer minimal documentation. This dissimilarity of upfront work is supported by Beyer et al. [2004] and by Fox et al. [2008], who broaden the difference to how the two methodologies allocate resources for requirements gathering.

Sy [2007] writes that the iterative nature of Scrum meant that UCD activities had to be completed in the same timeframe. In practice small designs could be completed in the 2-4 weeks a cycle lasted, but more time was required for complex designs. Their solution, to break the larger, complex design cases into smaller design chunks generated a further question - how could multiple smaller pieces for different designs be validated and investigated simultaneously?

Seffah and Metzker [2004] discovered also other challenges and notions regarding integration of UCD and software engineering in their research, such as

- Separation of software engineering (SE) and UCD processes. If the integration of SE and UCD processes remains unclear, it can lead to the UCD toolbox being regarded as dispensable and a development team skipping using the UCD tools in case of tight schedules
- Lack of a learning strategy when establishing UCD in an organization. Simply adopting an organizational model that supports UCD is not enough, as an applied learning strategy should reinforce continued promotion, evaluation and improvement of UCD methods in the company's software development process
- Companies not communicating the best practices of a team to other teams. In worst case scenarios this could mean losing that knowledge should a key person leaves the company
- Companies re-inventing the wheel instead of using already proven methods and methodologies.

Even though these are recognized issues, the agile literature does not determine a role for UCD, meaning that UCD must justify and define its

own role in software development [McInerney and Maurer, 2005]. Table 2.2 presents a summary of the challenges presented above.

Multiple definitions of usability	Difference of the suggested amount of upfront work
Difference of how resources are allocated in UCD and Agile development	Simultaneous validation and investigation of multiple designs
Separation of the software engineering and UCD processes	Lack of learning strategy
Separation of the software engineering and UCD processes	Lack of learning strategy to reinforce continued promotion, evaluation and improvement of UCD
Difficulties of communication the best practices of one team to other teams	Reinventing the wheel instead of using proven methods

Table 2.2: Summary of challenges considering integration of UCD and Agile software development

2.2.3 How they fit together

Despite the differences and difficulties in integrating UCD and agile, there are similarities as well. These consist of

- both UCD and agile development being iterative processes [Chamberlain et al., 2006; Fox et al., 2008],
- both encouraging user participation throughout the project[Chamberlain et al., 2006; Fox et al., 2008],
- and both emphasizing the importance of team coherence [Chamberlain et al., 2006]

Furthermore, an argument can be made that agile and UCD methods complement each other and thus create more value together - the gaps in the requirements engineering of agile methods as recognized by Düchting et al. [2007] and detailed in chapter 2.2.1 can be mended with UCD methods, and it could be thought that the heaviness of UCD methods could be balanced by the swift and iterative nature of agile methodologies.

The findings of Chamberlain et al. [2006] and Fox et al. [2008] are further reinforced in the research done by Ferre, who (as cited in [Düchting

et al., 2007]) determined that the basic conditions for integrating software engineering and usability engineering are an iterative approach and active user involvement.

As a final note, McInerney and Maurer [2005], who completed three case studies of how a team could combine UCD and agile, reported that even though, at first glance, UCD and agile seem hostile towards each other, all the UCD practitioners' reports in the studies were positive.

As UCD brings more knowledge of the user to agile projects, so can agile also be seen as beneficial to UCD. As Sy [2007] writes, in their work they have adopted a just-in-time design method, meaning that the UX team can focus on the few most important designs at a time, instead of the design for the whole release cycle, just as the agile development teams focuses on only a few new features at a time.

2.3 Methods for combining UCD and Agile

In this section we introduce and describe ten methods that have been used in academic research or by companies to combine agile and user-centered development. The methods were chosen based on the number of citations and how well they appeared to answer to the needs recognized at Napa, which are detailed in chapter 4.

2.3.1 One Step Ahead

"Key to our success as interaction designers on Agile teams is that we keep ahead of development, feeding a steady stream of designs into the Developer Track." [Sy, 2007]

When Autodesk adopted Agile development, the UX team found the initial tools offered to be insufficient to their design process, which relied heavily on observing detailed user behaviour.

Before moving to agile development, the UX team had some difficulties timing UX investigations. Their process then consisted of performing contextual inquiry, with rapid iterative design, before or at the onset of a project. In practice, development began at the onset of the project, meaning that design decisions had not been made yet when some features were already being implemented. These timing issues led to the UX team working almost a full release ahead, but this, in turn, led to out-of-date specifications that went unused.

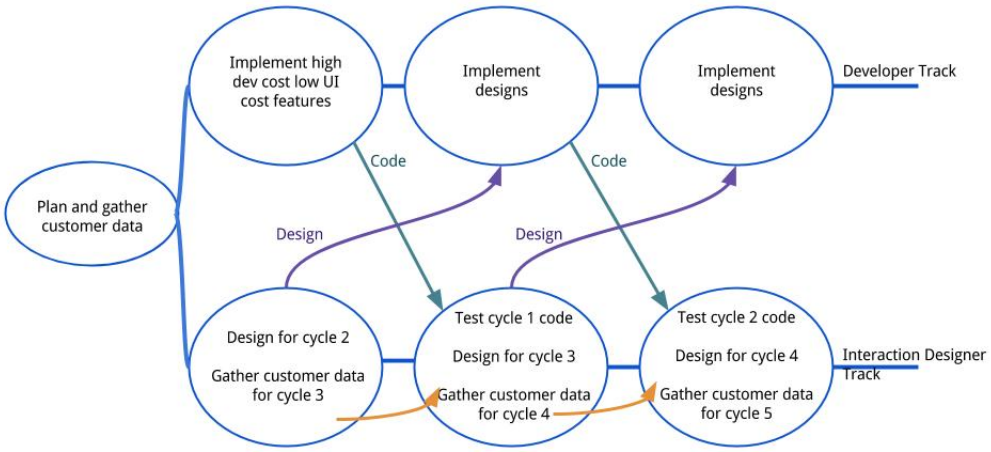

Figure 2.3: Parallel tracks for design and development [Sy, 2007]

After adopting agile development, the development team focused only on one feature at a time, giving the UX team the opportunity to not have to work with all designs in a release simultaneously. The key in integrating agile and UCD was, in their experience, creating two parallel tracks - Interaction Designer Track and Developer Track, which can be seen in 2.3.

Activities the UX team performed in cycle zero included data gathering, conducting contextual inquiries and exploratory designs for market validation, analyzing and summarizing prior data, and creating descriptions of users and workflows, among others. The consecutive cycles consisted of phases such as presenting designs from the previous cycle for development in this cycle, designing prototypes for the next cycle, and conducting inquiries and interviews in preparation to the cycle after the next.

One of the challenges Sy [2007] encountered while attempting to integrate UCD and agile was the granularity of the UX problems undertaken for investigation. Small issues could be research and solved in the time of one sprint, but larger problems required multiple cycles to solve. Their solutions was *design chunking*, where designs are broken into cycle-sized chunks. The

method was based on agile development, and in practice, this meant that smaller mini-designs were created, and these built upon each other incrementally. The chunks were subject to usability tests, and only the finished, complete design was given forward to development.

Another change they undertook was making communication lighter, by using *Design Cards* and *Issue Cards* to communicate UCD and UX plans and needs, and creating more traditional documents only for themselves, as the UX team had always been the main audience of these documents. Realizing that the UX team was the primary reader of the UX documents allowed the team to write far more concise documents than in the waterfall model. More of the Cards and documentation can be found in their research [Sy, 2007].

In short, the UX team now prefers using agile UCD for the following reasons, among others:

- More design is completed than before, and usability investigations are performed throughout the product's release lifecycle.
- Most important designs have priority, and no unused designs are created
- Changes suggested by, e.g., usability test results can be implemented in the current release

2.3.2 Lean UX

"Lean UX is defined as an approach for an extremely fast user-centered software development" [Liikkanen et al., 2014]

Liikkanen et al. [2014] write that the ultimate goal of Lean UX is to produce a product that satisfies customer needs as quickly and with as minimal resources as possible. Combined of the design thinking movement, Lean startup methodology and Agile software development, Lean UX is defined as an approach particularly suited for startups creating new products.

As the philosophy aims for quickness and minimizing the waste of resources, Lean UX discourages lengthy specifications and development processes. The fifteen principles describing Lean UX emphasize the quickly beginning to build prototypes and the continuous involvement of the users during the development. The six key principles of the Lean Manifesto, inspired by the Agile Manifesto, are as follows:

1. "Early customer validation vs. releasing products with unknown end-user value

2. Collaborative cross-functional design vs. lonely hero design
3. Solving user problems vs. adding cool features
4. Measuring key performance indicators vs. undefined success metrics
5. Applying appropriate tools flexibly vs. following a rigid methodology
6. Nimble design vs. heavy wireframes or specifications" [Liikkanen et al., 2014]

In the Lean UX book ideas are provided regarding the integration of Agile processes, such as Scrum, with the new Lean UX process. In regards to Scrum, during a two week development sprint user validation is done at the end of both weeks, and the design is revised based on the test feedback. In larger scope, sprints are connected to each other via themes, and each theme can last for several sprints. When a new theme is taken under development, it begins with sketching and ideation exercises which create the frame of the design.

The issue of heavy user involvement taking time and thus occuring only in the late stages of the Waterfall process has been solved in Lean UX by replacing the former massive user involvement with smaller sessions that are run more often and consistently. Users are part of the development process through e.g., user testing, and through being portrayed as proto personas. The proto personas are a deviation of the more traditional personas, in that after the hypothetical personas are built they are validated through interactions with users taking part in tests. [Liikkanen et al., 2014]

2.3.3 Discount Usability

"Even though the usability methods we used in the SunWeb project were cheap and quick, they were invaluable." [Curtis and Nielsen, 1995]

Curtis and Nielsen [1995] made a case for using usability engineering already in 1994, and in the article describe their light-weight usage of usability cards. In the case Nielsen worked on, with another designer, had a tight schedule for delivering the design, leading him to use discount usability, a method he had created some years previously. The product the team was designing was the internal WWW-information system for Sun Microsystems.

In the span of two weeks four usability tests were held, at various design stages and with different users at all stages. The first method used was Card Sorting, in which users sorted 51 on cards, each with a one-line explanation

of one kind of an information service that could be provided over the system. At the end of the test the user had divided the cards into groups, then combined them to form larger groups, and finally given names to the groups. Analyzing the raw data produced a list of recommended groups of features and suggestions for names of the groups.

The second phase was even more simple: based on the results of the first phase, 15 first-level information groups were defined and icons designed for them. After, new test participants were asked what the icons, with labels removed, represented. In the third phase, icons that had appeared unclear in the tests were redrawn, and a new set of test participants was introduced. The icons were magnified and printed on paper, placed on a table in an approximation of the layout of the home page, and the users were then asked to distribute the previously created cards in the most relevant area.

The final usability test, the Walkthrough, a magnified image of the design for the home page was printed out, and users were asked to point and describe what information they thought they could access through each button.

2.3.4 Design Studio

"With developers, stakeholders and designers in attendance, the design studio is an excellent forum for some ad hoc education on design principles in general, and UCD in particular" [Ungar and White, 2008]

In their research Ungar and White [2008] merged UCD into agile through a one day Design Studio, in which a UX team and the development team worked together to form a design direction and a deliverable. Additionally, the design studio also allows the team members to take ownership of the product and design, and ensures that the members have enough of a shared understanding to begin developing the product. Ungar et al. divide the design studio into three phases: the research phase, in which the UX team does UCD work such as observations and interviews for an appropriate time, the design studio pre-work phase, where, in essence, the UX shares the knowledge gained through the research with other team members and all team members create 3-5 rough sketches based on this data, and the studio phase itself, where all sketches are presented, discussed and critiqued, until one design concept is formed. After the studio the development can begin immediately, as there is a shared understanding of the design concept, and the UX team begins to produce mockups, wireframes and story boards based on the studio decisions and findings.

2.3.5 Rapid Contextual Design

"This process incorporates the customer voice and provides room for UI and user interaction design as part of the agile process."[Beyer et al., 2004]

Derived from Contextual Design (CD) by Holzblatt and Beyer, Rapid Contextual Design (Rapid CD) aims to be the fast, effective and customer-centered method to be combine CD with agile Beyer et al. [2004]. Rapid CD, according to Holtzblatt et al. [2004], does not mean doing all of the CD techniques rapidly, but rather aims to answer these questions:

- Do all of the steps need to be taken?
- When can I skip a step?
- How does CD fit into an existing design process?
- Could CD techniques be used to obtain user data and then already-familiar techniques be used on the data?
- Can a two-person team use rapid CD?
- What can be done in a few weeks time?

They go further on to determine, that "User-cenetered design will be seen as "rapid" if it can fit within the existing structures, expectations, and development processes od the organizations that deliver systems and products."[Holtzblatt et al., 2004]

Techniques of the contextual design are:

Contextual inquiry: Field interviews, observations and inquiries into work practices

Interpretation sessions and work modelling: Key points (affinity notes) capturing, analysing interviews, work practice model creation

Model consolidation and affinity diagram building: Data from individual customers is consolidated, and an affinity diagram is built based on all created affinity notes.

Personas: Personas should be collected on field data, collected from multiple users, and they help to communicate users' needs by bringing them to life.

Visioning: Team reviews all models created and invent how the system works, via hand-drawn sketches. "The vision represents the big picture of what the system could do to address the full work practice."

Storyboarding: Hand-drawn pictures and texts are used to picture the new designs for work tasks.

User Environment Design (UED): A single representation is created to show all the functions of the system and how they are organized in the system.

Paper prototypes and mock-up interviews: Actual users are used to test UI designs, drawn on paper.[Holtzblatt et al., 2004]

2.3.6 U-Scrum

"We have observed that the usability of the products developed under U-SCRUM is significantly higher than in previous effort." [Singh, 2008]

The U-Scrum method created by Singh [2008] takes a unique approach and bases combining UCD and Scrum on having two Product Owners, one in charge of traditional PO responsibilities and the other of usability and UX. The case presented in the paper starts with the POs working together to form a user experience vision. The usability-PO visited and observed several internal and external users, sought their vision on which tasks took the greatest amount of time to complete and then incorporated the needs of various stakeholders, such as internal customers, developers, and external customers into the formulation of the method. The complete vision was then presented to the team.

2.3.7 Product Canvas

The Product Canvas, as seen in 2.4, is a method and tool created by Pichler [2014] that, while not a scientifically proven method, could provide to be valuable in clarifying the big picture. The core of the method is to gather various aspects of the product development, such as personas, product functionalities, epics, and goals into a visual form. The method is divided into three main sections: Target Group information, Big Picture and Product Details.

The Target Group section consists of information about the users of the product. The Canvas method does not specify how the information in any of the sections is gathered, but suggests that personas, among other techniques, can be used in the Target Group section.

The Big Picture can contain such details as high-level visual design, user stories and journeys, product functionality, epics, scenarios, storyboards, constraint stories etc. - that is, everything and anything that can help the viewer to understand the big picture of the product. Again, these techniques are not

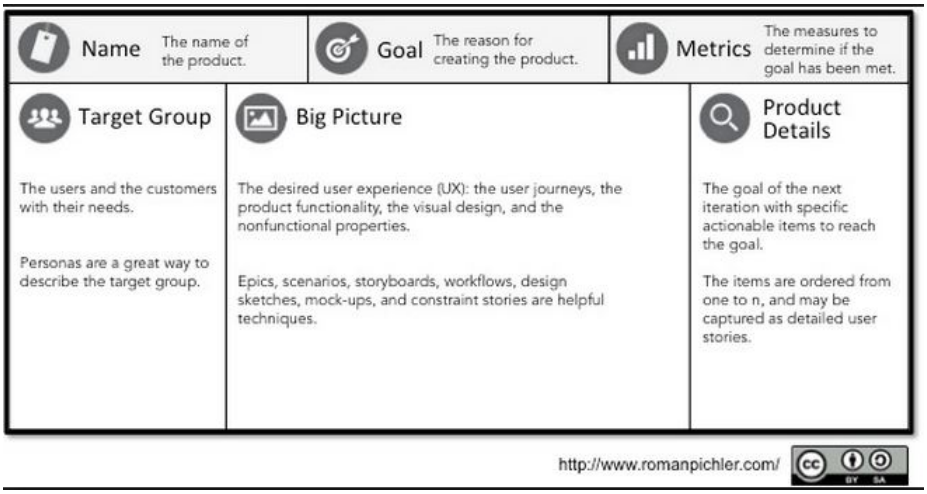

Figure 2.4: The Product Canvas [Pichler, 2014]

compulsory, but merely a suggestion on how the big picture can be gathered and formed.

The third section, Product Details, includes the goal for the next iteration and specific actionable items on how to reach that goal. The items in the section are the work to be done in the next sprint, ready stories, and details design sketches and mock-ups. Items in the Product Detail section are also ordered from one to n, and they may be captured as detailed user stories.

As there are no papers written on the tool, we do not take the tool at its face value, but still consider it interesting and promising.

2.3.8 Google Venture's Product Design Sprint

"The sprint is a five-day process for answering critical business questions through design, prototyping, and testing ideas with customers." [Google Product Design Sprint]

The Product Design Sprint, as described by is not as such a tool for combining UCD with agile development, but rather a five-day sprint, created with especially startups in mind, which aims to either accelerating a project or getting stuck projects moving again. Written by Google Product Design Sprint, the sprint can be divided into six parts, each with an agenda of its own.

0. Before the Sprint: Prepare
   Before the sprint begins the sprint organizer should make sure the issue

that the team tries to solve is a valid one as well as make sure, that the team taking part in the sprint includes all necessary roles and persons, such as designer, CEO, product manager and user expert. Additionally the organizer should also schedule a user study and find a facilitator for the sprint.

1. First day - building understanding
   Tools for this mentioned by the GV are looking at competitors products, interviewing personnel around the company, looking at existing research about the customer or the product, having the appropriate person talk to the team about the business opportunities, walking through the product as a user would, and looking at any analytics the business might have. After a shared understanding is built, the next step is to sketch the most important user story. The most important story is chosen by a group discussion and the sketch should simply show the user flow. Choosing the most important issue is influenced greatly by the problem the team aims to solve.

2. Second day - sketching
   The day begins by the team looking at the user story drawn on the previous day and evaluating whether the story should be divided into smaller parts. Odds are that it must be. Following the decision to split the story or not, the team members spend 15-20 taking notes and forming a mindmap based on all of the questions and notes raised on the first day. The third step, the Crazy Eights method, consists of sketching eight sketches in five minutes, creating a 3-piece storyboard based on the quick sketches and the mindmap, and then anonymously posting all of the storyboards for everyone to see.

   GV suggests a three-fold critiquing system: first, sticker based, where team members put stickers on ideas that they like to produce a heat map. Then, each storyboard is gone through and discussed, and lastly, a super vote, where members get two stickers they can give to the ideas they think are the best. If necessary due to company policy, team culture or other circumstances, appropriate sprint team members can have more stickers to give them more power in the voting process, or the vote can be completely skipped. The whole process is then repeated for all the user story parts.

3. Third day - deciding
   At this point in the sprint there are a lot of ideas and solutions, and the next stop is naturally deciding which of those ideas to pursue. Before

delving deeper into the deciding processes GV mention the possible need to combat the group effect - in a five day sprint setting the decision makers can act more democratically than they normally would, which can cause problems later on. GV suggests taking the decision maker out of the team room and asking for their opinion there, without the pressure of the team present in the room.

First thing to do according to GV is to see if there are any conflicts - meaning, if there are two or more solution ideas to a single issue. Noting these issues will allow the team to truly think about the various possible solutions instead of going simply with the first one that pops to mind. Otherwise, GV encourages to think about what kind of a user study the team wishes to do - one that compares a few different solutions, or one that delves deeper into the applicability of one solution. GV calls this "best shot or battle royale".

The other issue to consider during day three is what assumptions the team makes or has made about the business, the users, functionalities of the product etc. Writing these assumptions down and then figuring out a way to test them in the user study is a quick way to validate them. The final part of day three is dedicated to drawing the prototype, meaning that the team will toghether figure out what exactly will happen at each point in the user story and how this reflects upon the prototype the user will use and what they see in the prototype during various phases and steps. The end-result is a whiteboard "comic book" of how the prototype will look and act

4. Fourth day - creating the prototype
   During the fourth day the prototype is created based on the whiteboard user story made during the third day. Though GV does not say who it is that creates the prototype, the underlying assumption is that the UX, UI or some other appropriate person is in charge of this phase. GV advertises for simple prototypes with Keynote or PowerPoint, which do not require much knowledge to create. In case there is much to do and too little time, GV encourages to look to the team and see who could assist in creating the prototype(s). The key thing to remember here is that the prototype does not need to be genius level - a designer can always polish it afterwards. Other techniques GV mentiones are build an asset library that includes user avatars, formatted texts, screenshots and other details that other designers might need using a timer to make sure everyone knows how much time is left and can better schedule their work. Also timeboxing. Using the threat

of embarrassment by appointing a person to publicly point anyone out that is slacking from the job - according to GV simply the threat of this keeps people working. Giving quick critique at, for example, noon to keep consistency and get new eyes to see the prototypes having an outsider to review the proposal(s) If a keynote or PowerPoint prototype is not enough, but a prototype that requires actual code is necessary, GV suggests emphasizing to the programmer that the code is throwaway and does not need to be perfect

5. Fifth day - user study
   The study should be run with 4-6 persons the team hopes are their potential users - so not with anyone who works in the company. The blog post is directed more to the team members observing the study than the person conducting the interviews. First thing to do is to gather the assumptions made earlier in the sprint and list all key questions the team wants answered. It is also important to keep in mind that the user study is not a usability study, but rather find out what they think about your idea and prototype(s), what competitor products they might use, what kind of needs they have etc.. GV suggests making one person responsible for writing out one interview word-by-word as the study proceeds, and mentions that they do not always even take a video recording of the studies.

Though the design method is from the Google Venture's website, we consider the source trustworthy. The method itself is interesting, but as it has not been evaluated academically, we do not presume it to be a proven method, and before further evaluation too much emphasis on it should be avoided.

2.3.9 Hybrid, Generalist, Specialist

Our findings, combined with existing work show that the existing model used for Agile UCD integration can be broadened into a more common model. In this paper we describe three different approaches taken by our participants to achieve this integration. We term these approaches the Generalist, Specialist, and the Hybrid approach. [Fox et al., 2008]

The research conducted by Fox et al. [2008] involved members of teams performing agile methodologies integrated with UCD practices. Ten persons were interviewed from the teams, and three approaches to the integration of agile and UCD were identified.

The specialist Approach

The Specialist Approach consists of three member groups, the users customers, the development team, and one user-centred design specialist (UCDS), and two stages, the initial stage and the iterative stage. In the Initial Stage the UCDS conducts contextual inquiries to learn the basic requirements of the customers, and then creates low-fidelity prototypes. The prototypes are created iteratively, finally creating a an initial high-level UI design. In this stage the development team is almost completely absent, causing the UCDS to act as a "bridge role between the developers and the customer", relaying information of the customer's needs and requests to the development team. After the initial design is completed, the UCDS meets with the development team to discuss the technical feasibility of the design. If the design is accepted, the Initial Stage is considered complete, and the design is passed to the development team. The process is visualized in 2.5.

The Iterative Stage is similar to Sy's in that while the development team implements features for the current iteration, the UCDS conducts usability testing and further contextual inquiries for the next iteration.

After the technical design and implementation is completed, the development team passes the feature to the UCDS, who, in turn, passes in on to the customer or user for usability testing. If the feature is issue-free and accepted by the customer or user, it is marked as complete and the next iterative stage begins with a planning meeting.

The Generalist Approach

The Generalist Approach differs from the Specialist Approach mainly by the roles that developers need to practice, as the main roles in the Generalist Approach are the users/customers and the developers. This means that the developers involved in the research also had to act as UCD specialists. Even though the developers did not have formal training in UCD, they did have either some informal or self-taught UCD expertise. In some teams all of the developers acted as UCDSs, and in others only some developers were considered as UCDSs. In distinction to the Specialist Approach, all teams had at least two UCDSs.

The activities the developer-UCDSs performed were a combination of low-fidelity prototyping, contextual inquiry and usability testing, the differences manifesting in how long each activity was performed, and at what stage.

Once the initial contextual information had been collected, the teams either worked with the customer to create low-fidelity prototypes iteratively, or, alternatively, the team members acted as customers to

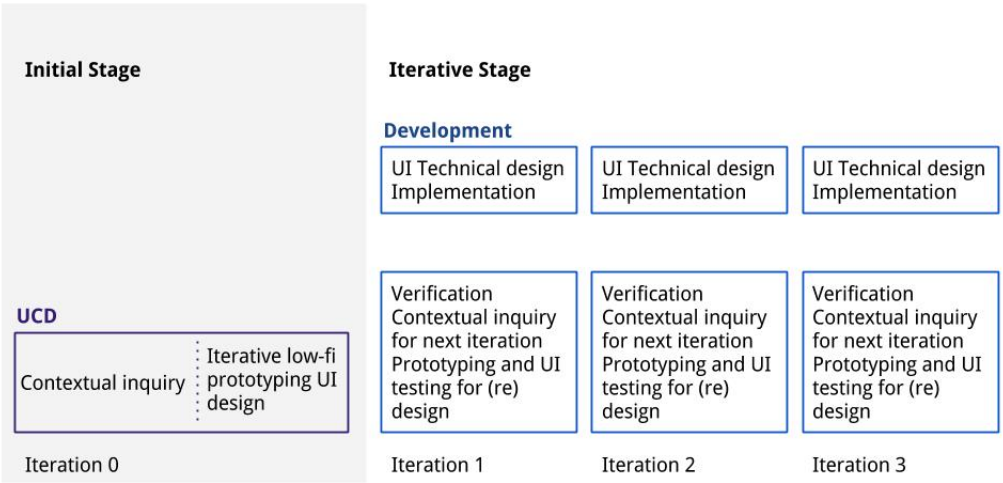

Figure 2.5: The Specialist Development Process[Fox et al., 2008]

determine stories for the development iteration. Furthermore, each team member was expected to take part in the usability testing in case the customer was inaccessible.

The UI design is considered completed after the initial low-fidelity prototypes are tested, and a planning meeting is utilized to prioritize features for the next iteration, and to split the work between developers. The Iteration Stage then begins with the completed initial design given to the developers for implementation.

After an iteration is completed, users are asked to complete a usability testing model, and developers watch the performance. Due to the developers acting as UCDSs, they can immediately grasp onto any observed usability issues without having to pass it on to another team member. Work schedule wise this means that developers work in parallel, some designing the UI and others implementing features.

Data suggest that the Generalist Approach is less formal than the Specialist, one possible reason for this being the way the UCDS team was introduced in the Specialist Approach. Some UCDS experts taking part in Specialist Approach teams corroborated the theory by reporting feelings of needing to prove oneself to the team, and by recounting feelings of "us and them".

The Generalist Specialist

The third identified team formation, the Generalist Specialist Approach, included user/customers, developers, UCDSs and a hybrid team member, formally trained in UCD with also software development experience.

The process of delivering software was similar to the other two Approaches, with Initial and Iterative Stages, and the main difference was in the role of the hybrid team member. Though capable of doing both UCD and software development work, both hybrid persons involved in the research acted more or less as liaison between the UCD team, developers and the customers.

2.3.10 Five Principles

"User-centred-design and agile methods are compatible, and they can work together but they can also provide problems if the key principles aren't addressed." [Chamberlain et al., 2006]

Chamberlain et al. [2006] investigated three project teams in an organization, with the aim to study how agile methods are used alongside UCD.

First, themes that appeared significant to the teams were identified, and in the second phase the themes were used as a framework for more exhaustive research.

Based on the themes and observations Chamberlain et al. [2006] created five principles to be acknowledged when considering the integration of UCD and agile methods:

1. User Involvement - the user should be involved in the development process but also supported by a number of other roles within the team, such as having a proxy user on the team.

2. Collaboration and Culture - the designers and developers must be willing to communicate and work together extremely closely, on a day-to-day basis. Likewise the customer should also be an active member of the team not just a passive bystander.

3. Prototyping - the designers must be willing to "feed the developers" with prototypes and user feedback on a cycle that works for everyone involved.

4. Project Lifecycle - UCD practitioners must be given ample time in order to discover the basic needs of their users before any code gets released into the shared coding environment.

5. Project Management - Finally, the agile/UCD integration must exist within a cohesive project management framework that facilitates without being overly bureaucratic or prescriptive.

2.3.11 Ten Usability Heuristics

"They are called "heuristics" because they are more in the nature of rules of thumb than specific usability guidelines"[Nielsen, 2005]

The ten heuristics defined by Nielsen [2005] are, in essence, a list of details any person can use to evaluate the usability of a product. Though expertise naturally brings more skill to the evaluation, we see the heuristics as an easy tool for even laymen to use. The list consists of the following items:

1. "Visibility of system status", meaning that the users should always know what is happening.

2. "Match between system and the real world", meaning that words, phrases, symbols and so on should be familiar to the user, and that information should be portrayed logically and naturally.

3. "User control and freedom", meaning that users should always have undo, redo, and emergency exit functions at their disposal.

4. "Consistency and standards", meaning that platform conventions should be followed in order to avoid confusion.

5. "Error prevention", meaning that design that prevents errors should be given more thought than simply presenting even well-thought error messages

6. "Recognition rather than recall", meaning that the user's memory load should be minimized by making actions, objects and options visible, and providing instructions of use in a reachable way.

7. "Flexibility and efficiency of use", meaning that users should be allowed to create shortcuts, and care should be taken to take both the novice and expert user into consideration.

8. "Aesthetic and minimalist design", meaning that irrelevant or seldom needed information should not be shown.

9. "Help users recognize, diagnose, and recover from errors", meaning that error messages should contain only a suggested solution, the needed information, and they should be written in a plain language.

10. "Help and documentation", meaning that help and documentation should always be easily reachable. [Nielsen, 2005]

Though the heuristics are from a web page, the source is the official web site of the Nielsen Norman Group, founded by Jakob Nielsen and Don Norman [Nielsen, 2008]. Thus, we consider the trustworthiness and value of the source and method to be large.

2.4 Common themes of introduced methods

Common themes recognized from the introduced methods are:

Methods in which designers working with users
: Tools and methods in which designers work with users by, for example, interviewing and observing them, and asking input from users during design are Sy [2007]; Curtis and Nielsen [1995]; Ungar and White [2008]; Holtzblatt et al. [2004]

Methods in which developers work with users

None of the addressed tools and methods specifically mentioned developers working in collaboration with users.

Methods that change development or design process

In [Sy, 2007] the design of a product is divided into smaller *design chunks*. These chunks are then researched, designed and verfied in the sprint schedule. In Lean UX, the traditional massive user involvement is replaced with smaller sessions with users, and the sessions are held more often [Liikkanen et al., 2014].

Methods that include sketching in the whole team

In both [Google Product Design Sprint] and [Ungar and White, 2008] all team members take part in sketching either the product ([Google Product Design Sprint]) or the design concept ([Ungar and White, 2008]). In Rapid CD [Holtzblatt et al., 2004] for example visioning, which includes sketching, can be used if found helpful or necessary.

Methods that exploit prototypes

Prototypes are used by Sy [2007], and are included in the list of contextual design methods of Rapid CD by Holtzblatt et al. [2004]. Prototyping is also used in the Google Design Sprint [Google Product Design Sprint] and in Lean UX [Liikkanen et al., 2014].

Methods that consider the management or personnel roles of software development

In [Singh, 2008] there are two product owners, one focusing on UX and the other on more traditional PO tasks. Fox et al. [2008] determined three kind of role-based approaches to UX - the specialist, the generalist and the hybrid.

Methods that aim to ensure big-picture understanding

The Product Canvas by Pichler [2014] composes a myriad of information of a product or a service into one, easily readable paper canvas. The five principles of [Chamberlain et al., 2006], on the other hand, provide a more high-level list of things to consider when attempting to integrate UCD and software development. Finally, the Rapid CD of Holtzblatt et al. [2004] again contains methods, such as the User Environment Design, that can be used to better understand the big picture of a product.

Chapter 3

Methods

In this chapter we first go over how the state of user-centered development at Napa was researched, followed by how the gathered material was analyzed. We then review the methods described in chapter 2.4, evaluate their applicability to Napa and describe which methods we chose to test, and finally describe the tool we created based on the interviews with the developers.

3.1 Investigating the state of user-centered development

Answering the two research questions, "how can UCD be introduced and integrated into Scrum development in a company with very few or no UX professionals" and "what kind of UCD and UX tools would developers find valuable when developing software?" required us to profoundly understand the state of UCD at Napa and in the chosen teams. To gain this understanding, we organized initial interviews with team members and the POs of the teams. One-on-one interviews to discover the needs of users are encouraged by, for example, Cooper et al. [2014] and are a tool of the Rapid Contextual Design [Holtzblatt et al., 2004].

Three developers and the Product Owner (PO) were interviewed from the Safety team, and, similarly, three developers and the PO of Voyage Optimization team (VO). The interviews were semi-structured, with the PO interviews differing slightly from the developer interviews. All first-phase interviews lasted approximately 40-60 minutes and were recorded. The interview questions were put to order on an excel sheet and answers were gathered to it during each interview. Later, the recorded interviews were transcribed roughly and answers collected during the interviews were complemented by findings made during the transcription process. The semi-structured inter-

view layout of the first phase interviews can be found from appendix A. The interview focused on three main areas - getting to know the interviewee, how they see the big picture and its formation in their team, and how users are involved and taken into consideration during development.

After the initial interviews were completed and the recordings transcribed, the transcriptions were reread to find recurrent themes and issues. These themes and findings are examined in detail in the next chapter.

At the end of the research period the same three developers were again interviewed from the Safety team, but only two developers from the VO team, as the third one had left the company. The final interviews lasted approximately 10-30 minutes and were recorded. Similarly to the first-phase interviews, the questions were written in excel and answers gathered to the sheet during the interviews, and the recordings roughly transcribed. Answers gathered during the interview were complemented by realizations done during the transcription process. The semi-structured interview layout can be found from appendix B. The final interviews focused on four areas: how the interviewee felt about the workshops and the research, how the workshops could have been improved, how the research impacted the interviewees feelings regarding UCD and UX, and what had been found useful (or not) in the workshops.

In addition to these team interviews, two developers from the Hull team were also interviewed at the beginning of the research, but after the initial team interviews, with focus on their experiences with working with the hired UX consultants. These interviews were also recorded, but due to technological difficulties the quality was very poor and the recordings could not be transcribed. Notes were taken during the interviews, and only the notes were used when constructing our research. The themes recovered in these interviews are also gone over in the next chapter.

Finally, during the research period multiple informal interviews were held with various personnel, and no transcription was made of these. Elliott and Jankel-elliott [2003] state that "much of the richest data which ethnography can capture comes from the whole realm of informal talk between researcher and informants". The informal interviews during this research happened by the coffee machine, company gatherings, and during lunch, and did not have a set of predetermined questions, but rather discussed the initial interview topics further. As Elliott and Jankel-elliott [2003] continue, "the essence of the informal interview is that the researcher does not have a written list of questions but rather a repertoire of question-asking strategies to select from when the moment seems appropriate".

3.2 Introducing and integrating UCD

In order to facilitate conversation, introduce the tools and chosen UCD methods, and to observe how the teams reacted to these we organized workshops and facilitated them. The facilitated workshops were not a research method as such, but a natural way of bringing a team to the same room to discuss the topic, as we wished to focus on group dynamic and conversation rather than individual thoughts at this point.

Developed originally for use outside of software engineering domain, facilitated workshops are not a new method, and multiple books have been written on them [Kähkönen, 2004]. In their research Kähkönen [2004] conclude that "Facilitated cross-team workshops, while a simple and intuitive practice, were perceived to be an effective practice for cultivating knowledge in these communities".

Some examples of workshops include the "Google Glass Development in Practice: UX Design Sprint Workshops", whose authors write that "similar workshops can become a breeding ground for new ideas" Wichrowski et al. [2015] and the Inspiration Card Workshop of [Halskov and Dalsgård, 2006], who describe their method as a collaborative way to create new concepts for design though combining findings from domain studies with sources of inspiration from technological applications.

3.3 Current state of UCD at Napa

In this chapter we will first introduce what the current development process is at a high level at Napa, and how UX is seen at the board level. We will also briefly show what steps have been taken in the company to introduce UX, and analyze these measures. In addition to this, we include descriptions of what kind of development processes the teams that were involved in the research have, and how UX was seen in them before the research.

3.4 Corporate level

The current development process at Napa consists of (mostly) two week sprints and three-month long release cycles. At the end of a release cycle is a release day, which many attend, but is not compulsory. At the end of a sprint is sprint review, which includes the team.

Usability and UX have been recognized in the board as important aspects in the evolution of Napa products, but there haven't been many actual steps

taken to realize this. A UX consultancy team was hired to work part time with one team, Hull Finland, and a UX person was hired in fall 2014 after completing a trainee position at Napa.

Based on the interviews and discussions we have had, it appears that while there is a theoretical support from management to invest in UCD related activities, the need to focus on technology and to push out updates is still big enough that it is difficult for POs to look into UCD and gather applicable work methods and tools to try. At the end of the research period, however, we felt that the attitude was clearly shifting, and this was supported by, for example, the "Napa UX Day" event, which was organized in-house by Napa employees.

3.5 State of UCD at the Safety Team

In the Safety team, the general atmosphere in the interviews was positive towards users and user-centered actions. All interviewed developers said, that in case of developing a new product or feature, they'd like to know who they are developing for. One developer described, how "often in the release plan there are items that no one can say if they are good or bad".

When asked at what point user involvement would be most beneficial, two of the interviewed developers stated that user involvement throughout the whole project would be most beneficial. One of these developers iterated further, that involvement in the beginning of the project is most crucial, then the end, and involvement in the middle of the project would be least important. The third developer also spoke positively of user involvement, stating that "in the beginning it would be important that users participate in introducing their needs so that they will be clear to the developers. – Then we can better specify what needs to be done". In addition to these, one developer also mentioned prototypes as a potentially good tool for software development, and another spoke of Personas created in collaboration with PM.

On the other hand, the big picture of what the team was doing was not clear to the developers, and the communication flow was also found confusing. One developer stated that "the big picture does not show very well to our level from anywhere, you might find the roadmap information or you might not." He also described communication as problematic and like a "broken telephone", as all team members do not use Flowdock or Lync for communication. He stated that the Daily Scrum is the clearest routine communication channel.

Another developer categorized the team's communication methods as

such: Lync used for personal matters, Flowdock for more common topics, joint meetings with the team in India, but more communication inside the team room than in the aforementioned channels. He felt that while all "classic Daily issues" are gone through in the Daily Scrums, not all issues that should be discussed have emerged. The big picture, he describes, is a splintery entity formed of all the above meetings and communication channels - "maybe everyone then tries to form a big picture of what we are doing from those".

Wikipages were not mentioned by any of the developers, but the PO of the team acknowledged these as one tool to communicate the big picture. The customer of the team has also had access to some of these pages, but Confluence, which also wasn't mentioned by the developers, has been introduced as a new tool to replace the Wikipages. The Safety team creates a multitude of reports based on the customers' needs and specifications. One clear challenge in this is that the requirements of, for example, Asian and European customers are very different, and thus the need to customize a product is great. When asked what kind of tools or skills the PO would imagine the team could benefit from, he stated that "a better understanding of the end-users' work habits and processes would be immensely beneficial". Furthermore, he believed that the user involvement would be beneficial in all stages of software development.

3.6 State of UCD at the Voyage Optimization Team

In the Voyage Optimization team, VO, three challenges rose from the interviews: incomplete knowledge of the user, the lack of a big picture, and conflicting relationship with users and UCD.

Incomplete knowledge of the user manifests as the "team doing something and the user using it differently", as stated by a developer. This, in turn, can lead to issues for the user when trying to do his job, decrease of work efficiency, and lead further on to an unhappy customer. Not knowing the users can also mean that some critical factors are missed or overlooked during the requirements management and development phases, and business opportunities can remain unidentified.

The lack of a big picture manifests as difficulties to understand what is the issue that a feature, for example, is trying to solve. Collaboration between different departments (eg. marketing, development, customer service) is lacking, and no effort is done to evaluate if the product is working as it

should. One developers commented on this that "asking brings problems, so hush hush". One developer also said that requirement specifications are not always unambiguous, making it difficult to know what is the true user need and context for the need.

Finally, the contradictory relationship with users and user-related activities was very clear. On one hand, a developer first stated that "users don't know what they want or they just nod their heads", yet at the end said that it would be beneficial if users could use the product before it is release, so that bugs and errors can be found before the product becomes widely used.

Another developer was moderately positive of user collaboration, stating that they receive "regrettably little feedback, as it would be nice to know if the product works". The third interviewed developer anticipated that collaboration with users would be beneficial in some ways at least, but thought that the PO should be the one to be in contact with the users, and then offer analysed thoughts and results to the development team.

When specifically asked, tools that the developers wished were something to help portray and understand the actual steps taken by the customers, more visits to customer locations, Personas, and the aforementioned already-analysed information from PO, who has been in contact with customers. One developer also stated that they did not think developers would be very interested in methods.

The PO of the team explained that the team already gets feedback from customers through, for example, demos, which can be given to customers for testing also outside of the release cycle. Though he felt that while observing a user is useful, he stated that usually a contact person reachable by phone or e-mail is enough. In slight contradiction to this, he also stated that a better understanding of the end-users work habits and their processes would be immensely beneficial.

3.7 Hull team and Outsourced UX Consulting

Overview of collaboration

The Hull Finland team began working with a user experience consultancy roughly 1,5 years ago, and during this time the consultants have also collaborated with some other teams.

The methods they have introduced (as explained by the developers) are UX lab and personas, of which personas are not used at all, and UX lab is currently used quite irregularly. In addition to these they have provided and

assisted in creating a visual look for Napa and especially for Napa for Design. In addition to these the interviews developers stated that they feel that they think more of the user now, though they could not pinpoint any one reason for that.

The process of combining UI and interaction design into Scrum is still being developed. Some difficulties, as per developers, is the classic question of how much design can be done upfront, where should UX lab and/or user testing be situated schedule-wise, and what will design do while the program is being developed.

One wish made by a developer was that the consultants (usually two persons, working approximately two days per week at Napa) should work more with Napa, and more on Napa's premises. This sounds logical, as the scheduling difficulties could be solved more easily if there were more UX designers working on the project.

Interview results

Both of the interviewed developers had had a positive experience of working with the consultants, one developer stating that he felt his "approach to software development had changed" and the other saying that personally for him starting to think is the feature or product usable in the user's context. The most concrete benefit of the consultants joining the team, according to one of the developers, had been that "there is now a visual designer on the team", who chooses the colours and components. The other stated that the UX lab was the largest benefit, as it had allowed him to see that what they do actually matters. It was also seen as a positive that the consultants had spread some user-experience knowledge to other teams through collaboration, with mentions to a Loki team and the VO team.

Both developers had recognized some challenges, though, which we have divided into three areas: integration of UX and software development processes, outsourcing UX knowledge, and understanding UX.

Integration of user experience and software development was found to be still confusing, and one developer stated that "the process is constantly changing". The team had tried the method of design being one sprint ahead similar to [Sy, 2007], but had not been satisfied with it, as there had been conflicts with vacations, sudden changes to the backlog, and other, unspecified obstacles. One developer stated that there was an ongoing "the inmates are running the asylum" -situation, where the developers pulled for new features and feedback, essentially being leaders in what would be developed next, instead of the UX designers pushing for new products. The developer also expressed that the team seemed to have difficulties in finding

the balance in the development process, as the design had moved from one sprint ahead to be almost simultaneous with production, and there was fear that the amount of documentation would go up.

Furthermore, a developer thought that the responsibility of UX had not been well-divided, and felt that it was not understood how one person could not alone be responsible for the whole user-experience of a product. One developer also stated that no-one demands UX of them, so UX does not happen at the hands of the whole team, but rather those interested in UX practice it and others do not. Thus far the lack of clear guidance in UX matters had manifested as the withering away of the UX lab, and the Personas done by the consultants "being in a heap on a table one morning" but no go-through or guidance of how to use them.

Outsourcing was not seen as a problem as such, but the part-time work and turnover of consultants was thought to be problematic. One developer stated, that "many issues rise in the retrospective meetings" but the consultants do not usually take part in them. The turnover was also seen as a negative - while mostly the same consultants had consistently worked with the Hull team, the training of new consultants to understand Napa, its customers, and its products was seen as time-consuming.

Finally, the third challenges related to **Understanding UX**. One developer wished that there had been an overall introduction into what UX is and how it binds together into software development, and the other wished for an overview of the UX methods to be used.

New methods were introduced during the software development process, and according to one developer these methods included UX lab, UX measures and indicators, questionnaires, personas, and paper mockups, and none of these were used daily. The other developer mentioned only the Personas and UX lab, stating that if there were other methods, they must have been introduced very "ninja-esquely".

When asked what should be done differently in the beginning, one developer stated that the team in question should start to think about how to integrated UX into the development process immediately, as they had only recently started to consider how it should be accomplished. The other developer stated that the best way of educating persons with non-ship building backgrounds into Napa's products and customers was through work. Both developers also highlighted the UX lab and observation as good tools for software development.

Based on the interviews it seems that the benefits of working with the UX consultants are threefold: they have introduced new tools, such as the UX lab, they've been able to encourage the developers to think about users, and they have taken the responsibility of UI design from the developers.

What is still missing, though, is the routine in combining UCD and agile, and practical tools the developers could use in their design and development work. Furthermore, we find it slightly problematic on the whole, that the design services, which are needed in a varying manner and schedule in many teams, are currently outsourced. Resources are in this situation used on personnel who might not work for a very long time with Napa, and training own personnel will take time.

3.8 UX Maturity

We evaluate Napa to be somewhere between the stages one (beginning) and three in the UX maturity model of [Chapman and Plewes, 2014]

Stage 1: Beginning

1. "Any UX design activities have very little formal structure (developers do the necessary elements of screen layout) and are probably not considered part of a UX design activity.

2. There are no UX design goals tied to business objectives."

Both VO and Safety team developers regularly designed the user interface elements, and neither team had determined UX design goals.

Stage 2: Awareness

1. "There is little user feedback or it is limited to asking users their opinions on design or functionality.

2. UX design goals are general or hard to measure (for example, "the interface should be intuitive and straightforward").

3. A UX professional is consulted during a projects too late in the process.

4. There is inconsistent awareness and buy-in to making UX design investments, such as training, beyond a few people."

Neither VO or Safety consistently gathered user feedback of their product, and, as mentioned in stage 1, neither had UX design goals. A UX professional was, in effect, never consulted, and inconsistency of UX awareness could be found in both teams and in the organization (e.g., some team members knew more of the experiences of the Hull team, while some did not).

Stage 3: Adopting

1. "There is no senior leadership or management in UX. The UX function may report into Marketing, Product Management or Engineering or is distributed between individuals across projects.

2. There is no standard design and development process being practised across the organization. For example, some projects or parts of the organization may insist on usability testing of products, while others do not.

3. A common perspective on UX design does not exist throughout the organization. The expertise exists within the company at the project delivery level, not at the executive level."

With only one recently-graduated, just hired UX professional and some outsourced UX consultants, there is a clear lack of UX seniority. As mentioned above, the processes and best practises created in the Hull-team were not communicated consistently to all other teams, paving way fo inconsistent development processes in the future. Finally, the interviewed developers had varying attitudes towards UX and UCD, hinting that there is no common shared perspective throughout the organization.

3.9 Summary of Challenges

There is one clear lack in the process of introducing and integrating UCD at Napa, and that is communication. The experiences and processes developed in Hull Finland team are not communicated in almost any manner to the other teams. The Sprint Release meetings are the main communication channels, but attending those is not compulsory for anyone, and the possibility to share intimate details and feelings of UCD development are small as the focus is on presenting features and updates developed during the last release cycle. There are also scrum of scrums, but they have gotten mostly neutral or even slightly negative evaluations in our informal discussions with various people.

Combining the lack of feedback gathering with limited possibility to introduce and integrate UCD can easily result in software that does not take into account the varying needs of users of varying skill levels [Sohaib and Khan, 2010]. Lack of feedback can also lead to unknown bugs that the users have not reported forwards, which in turn can cause significant difficulties in future development.

These challenges are very similar to those listed in chapter two, as described by Seffah and Metzker [2004]: separation of software engineering and

UCD processes, lack of a learning strategy, and difficulties in communicating the best practices from one team to others.

In table 4.1 a summary of the discovered challenges is presented.

Safety Team	**VO Team**	**Hull Team**
Splintery communication through various channels and medias	Incomplete knowledge of the user	Scheduling of UCD and software development tasks
Lack of user involvement	Lack of user involvement	Sharing of success and best practices with other teams
Lack of understanding of the big picture	Lack of big picture	Some methods (Personas, UX Lab to some degree) simply not used
	Conflicting relationship with users and UCD	

Table 3.1: Key Challenges at Safety Team, Voyage Optimization Team, and Hull Team

3.10 Applicability of methods to Napa

Finally, here we summarize the applicability of the methods introduced in chapter two to Napa. As mentioned previously, in addition to the consultancy working at Napa, there is only one professional usability and UX employee. Therefore the methods that could provide value to the development teams must be light-weight and easy-to-use, to avoid the need to hire more UX professionals or to transform developers into UX experts. Thus, the methods we decided to introduce to the teams were Nielsen's 10 Heuristics, Personas, Product Canvas, and the Crazy 8's of the Google Design Sprint.

Nielsen's 10 Heuristics can be used as simply as reading the heuristics from a paper and using them to evaluate a design. Though expertise naturally brings more depth to such evaluation, we share the sentiment of [Curtis and Nielsen, 1995] "some usability testing is so much more than none". Personas are an efficient way to share knowledge of users - discussion is easily sparked with the Personas as prompts, and thus a shared understanding is hopefully created at the end of the process [Pruitt and Grudin, 2003].

As the Personas create shared understanding of a very specific subject,

the users, thus the Product Canvas has the capability to do the same on a more high level. Again, the information is collected in a very visible way, and collecting the details demanded by the Canvas does not require expertise of usability, UCD or UX. Finally, the Crazy 8's of the Google Design Sprint do not require any expertise on what makes a good user interface as their purpose is to create a multitude of ideas and sketches.

One Step Ahead, [Sy, 2007]	While the process introduced by Sy [2007] appears to be highly effective, it is not applicable to Napa. As there is no central UX team, nor a UX person to work with multiple teams, design and designer heavy processes, such as this, do not fit into Napa's environment.
Lean UX, Liikkanen et al. [2014]	
Lean UX could very probably assist Napa in combining UX and software development, but we believe it to be and demand a more comprehensive change than can be managed during this research. Additionally, it does not provide answers to "what kind of UCD and UX tools would developers find valuable when developing software".	

Discount Usability, Curtis and Nielsen [1995]	While these methods are light-weight, and relatievely easy to use, they do require the facilitator(s) to be versed in usability matters, and that there is a person or a team capable of creating UI designs and icons. For these reasons the methods are mostly not applicable to Napa. We do, however, feel that the Walkthrough provides interesting possibilities. A good, old-fashioned usability test is always valuable, but quick insights could also be gained simply by asking what the users think would happen if they pressed a button, or what information clicking on an icon could provide. The most important matter, though, especially regarding this research and the case teams, could be considered to be the last sentence of the article: "Even if your project schedule seems too compressed to do much usability work (ours certainly did), you can still do something, and some usability testing is so much more than none." [Curtis and Nielsen, 1995]
Design Studio, Ungar and White [2008]	We consider the biggest benefit of the studio to be the shared understanding that is gained through working closely together - information flow is better and all members have a possibility to take part in the design. Ungar and White name the time-compressed nature of Scrum to be the biggest challenge when implementing the Design Studio, we consider the lack of actual interaction between the developers and users to be a potential pitfall. While the applicability of the method as such to Napa is not possible due to the lack of a UX team, the concept of creating a design concept together, and in essence forcing all team members to think of the UI, UX, and users, is very interesting. We contemplate that one day spent away from development could very well prove to be highly beneficial, if, as a result, users would be taken more into consideration well before a feature of product is planned to be released.

Rapid Contextual Design, Beyer et al. [2004]	Even in its rapid form, contextual design requires educated UCD professionals and includes relatively much up-front work. As there Napa has no UX or UCD team, using rapid CD is not possible. Additionally, even though the methods of Contextual Design do provide good and deep insights, most of them are again too heavy for a UCD team without a UCD professional to complete, and educating the developers to do the tasks is not in our mind a preferable solution as the methods require much time to complete.
UScrum, Singh [2008]	As Singh writes, the interactions of the two product owners focus on three main artefacts: a project plan, a user experience vision for relating POs to the stakeholders, and a backlog for relating the POs to the developers. While structurally the same as a traditional backlog, the U-Scrum backlog includes more consideration for usability. This consideration is formed through higher priorities for stories that have a predominantly usability related impact and through greater detail in regards to the potential acceptance criteria for user stories, of which some refer to the personas. There is already similar division of responsibilities at Napa, with Product Managers being, in some ways, responsible for knowledge of the user and their needs. However, as we show in part four of this paper, the role definition is ambiguous, and who is ultimately responsible for a product's success is understood in various manners by the developers. While results and observations of the concept are highly anecdotal and more research is needed, we do feel that the U-Scrum is an interesting concept and could prove beneficial, at least for a more clear division of tasks and responsibilities. Nonetheless, as the focus of this paper is on tools for developers and methods for integrating UCD into agile, we do not research how this method could fit in at Napa.
Product Canvas, Pichler [2014]	While there is no academic research on the method, the Product Canvas very conveniently brings together various important aspects of agile and UC development, and makes the results visual.

Google Venture Design Sprint, Google Product Design Sprint	The attraction of holding a five-day-long design sprint is great, as it very likely would produce great results. Reserving five days for product design regularly, when much of the work done is updating existing products is not applicable, though, but the potential of the design sprint is too great to be completely ignored. Each day holds has an activity that is a response to the challenges the teams have faced: building understanding of the user and the context, sharing thoughts and ideas, and making decisions as a team, sketching out ideas and sharing those, and finally prototyping and using real users to test the prototype. Due to these effects we chose to test the Design Sprint, although on a much shorter timescale.
Specialist, Generalist, Hybrid, Fox et al. [2008]	For Napa, the Generalist Approach could prove to be valuable, as the current situation is almost identical in most teams. However, in order for the approach to function properly, all teams should have developers interested in UCD, and education should be provided for them. A further iterated approach, where the current UCD specialists could, for instance, provide assistance in planning contextual inquiry could be considered.
Five Principles Chamberlain et al. [2006]	While the five principles, *user collaboration, collaboration and culture, prototyping, project lifecycle, and project management*, do not offer much hands-on advice or tools on integration of UCD and agile, the principles still are something that any organization seeking for the integration of UCD and software development should consider. However, as the focus in this paper is on tools and methods for the integration, and tools for developers to use in UCD work, we will not at this point introduce these principles to the teams. Further on in the integration process, however, bringing these principles especially to the knowledge of the teams and company management is highly crucial.

Table 3.2: Applicability of methods, introduced in chapter 2, to Napa

Chapter 4

Implementation and Results of Workshops

In this chapter we first go through the tools and methods we implemented in the two teams in detail. Then, we describe how the methods were implemented in the teams, and go over the problems and challenges we met during implementation. For each difficulty we also suggest a solution.

4.1 The tools and methods tested

4.1.1 Sit down with colleagues

Though not a method as such, we view improvement of cross-team communication to be a large part of stabilizing UCD's position at Napa, as currently the good experiences of the Hull team with UCD are not communicated to other teams. To improve communication, we had a developer from Hull Finland team sit down with both Safety and VO team, and explain what they had done with the UX consultants, and what kind of results they have seen.

4.1.2 Personas

We chose Personas, as described by [Holtzblatt et al., 2004] in chapter two, to be one of the introduced tools, because the general opinion of "we know our users" was largely felt by many. This may, in many ways, be true - Napa is filled with highly educated ship engineers and professionals, who have worked on ships, and many of their customer-relationships are old. Engineers at Napa know the protocols and official steps used by their users without any doubt, but what is lacking is the understanding that without

full knowledge of the persons behind the protocols and the contexts the users work in, the knowledge, no matter how wide it is, is lacking. We believe that using and creating Personas is a good tool to combat this narrow knowledge of users, by widening it and making users seem more real. Simply working in a group when creating Personas should allow team members to bring their hidden knowledge forth, and help build a complete picture of a typical work environment.

Furthermore, we noticed that although some developers have a very thorough understanding of users, this knowledge is not actively shared with others in any way. This raises the question of how new employees will learn of the users, when there is no common and easy way to systematically share knowledge. Using Personas will help alleviate this issue, too, by being a "medium for communication" as Pruitt and Grudin [2003] discovered. When a new employee joins a team, handing them thoughtfully created Personas, and explaining them even briefly, could help spread knowledge gathered over the years.

4.1.3 Nielsen's 10 Heuristics

As one of the themes at Napa has been how to create better UX and usability in their products, the Nielsen's 10 heuristics [Nielsen, 2005] appear to be a good way of introducing the concept of usability and how it could be evaluated. Even slight and simple knowledge of usability will help the developers to consider the product or feature they are creating [Curtis and Nielsen, 1995]. As the 10 heuristics are easy to understand with only ten items, easy to remember and in our opinion do not require years of training to use, they could prove to be a valuable tool for the developers.

4.1.4 Crazy Eights

The Crazy Eights of the Google Design Sprint is a great tool to "crank out variations of ideas quickly" [Google Product Design Sprint]. Though it might push some developers outside of their comfort zone with the sharing of ideas and sketches, the usability of products and features coudl improve simply through the simple equation of more heads creating more ideas than one person. The Voyage Optimization team members especially mentioned how the developer is the one that creates a UI, and just maybe might share it with someone before implementing it. By using the Crazy Eights, the whole team can give their thoughts on what a good UI might look like, creating multiple potential solutions. Even if only two developers take part, there would still be 16 personal sketches and two shared sketches to evaluate and

evolve further. We do, however, suggest starting this slowly - sharing ideas and sketches can be felt as a very vulnerable process, and an exercise or two should be run before the tool is taken into full use. These exercises could be either for a old, current, or future feature, or they could be of something completely unrelated - e.g. a mobile application for shopping shoes. A completely unrelated exercise topic could help lower the threshold of sharing one's ideas.

4.1.5 Product Canvas

The Product Canvas [Pichler, 2014] was chosen because many comments we heard from the developers suggested that the big picture of a product is seldom clear, and, to some degree, seems to be controlled by no one. To enable a more shared understanding we suggest using a tool similar to the product canvas. Again, one of the goals is to share understanding and create a common understanding of the product to be developed. An additional benefit the canvas could bring is visibility - if the canvas is done on paper, and attached to a wall or some other highly visible place, then it would be easier for the team members to also remind themselves how a feature fits into a product, recap goals, and, all in all, keep the product level plans and visions in focus easier.

4.2 Implementation and Results of workshops in the Safety Team

Here we go over the two workshops organized with the Safety Team, and results gathered from them. Finally, we provide an observation and evaluation matrix of the methods tested.

4.2.1 Implementation of Safety Team Workshops

First Workshop

First workshop was held in December, when the team members located in India were visiting Finland. The workshop was held at the beginning of a regular Sprint planning session, and roughly 1 hour 20 minutes were allocated to the workshop. The agenda of the workshop was as follows:

1. What are UCD and UX
2. Personas

3. Nielsen's 10 Heuristics
4. Product Canvas
5. Crazy Eights

The methods were introduced as a power-point show, with printouts of the product canvas and the 10 Heuristics for each attendee.

After a brief introduction into the thesis and UCD, the focus moved to the tools. First, the concept of Personas Holtzblatt et al. [2004] was explained, and after discussion three archetypes of users were identified. The plan was to create Personas of all three types, but only one complete Persona was managed in the allotted time. The Personas were created on a flip chart, and the creation of the first Persona was facilitated by us. With some prodding all team members gave their thoughts and opinions on the Persona. For the second Persona a team member from India volunteered to facilitate, and although the discussion was still somewhat active, there was clear a decrease. Additionally, the creation of even the first Persona took so much time that the second Persona could only be created half-way through.

After the Personas were done, the level of communication decreased significantly. In the rest of the workshop and in the second workshop most of the communication was done by two to three people, PM, PO and one of the developers. We went over Nielsen's 10 heuristics [Nielsen, 2005], but this did not ignite much discussion.

During the final interviews it transpired that what most stuck with the team members was the creation of Personas, and from the second workshop, which was held much closer to the final interviews, not much. The team members who reside in India had given particularly positive feedback of the Personas, as it turned out that they had not known in much detail what the users of the product were like. This feedback was not given directly to us, but was relayed by the team's PO.

Second Workshop

One of the biggest challenges in the first workshop had been facilitating conversation, and this challenge became more apparent during the second workshop, which was held on Lync. We considered trying to organize some exercises on Lync, but decided against it as we felt it might only complicate the workshop. Thus, the power-point presentation we held was not interactive, and there was even less discussion than during the first workshop. In the second workshop a developer from the Hull Finland team also joined momentarily to tell of the team's experiences with UCD and the outsourced

UX consultants. Again, most of the questions came from one or two persons, and the discussion was relatively brief.

The second workshop focused much on recapping the items we had gone through in the first workshop - we deemed a Crazy Eights exercise unfit for a Lync meeting, and wanted to focus on the tools we felt could be valuable to the team. The contents of the workshop were:

1. Hull team's experiences
2. Personas - recap
3. Nielsen's 10 - recap
4. Asking the right questions

When preparing the second workshop we began consider in detail what the developers should know about the items they develop, and the result was the "Asking the Right Questions" segment. In this second workshop with the Safety team, the method consisted of the following questions:

1. Who will use this feature?
2. Why does he need it?
3. When does he need it?
4. Multiple times?
5. How does this feature help him?
6. What information/knowledge does he or the software need to know before using this feature?

After the workshop we worked the idea further, and eventually formed the UX Checklist, which we will detail later in this chapter.

4.2.2 Results of the Safety Team Workshops

The most important finding in the workshops was that there seems to be very little communication and shared understanding of, e.g., who the user of the product is.

The main challenge in both workshops was the lack of conversation between the team members. In any future workshops it should be carefully considered if there would be a more relaxed and open atmosphere with maybe

only the developers present, though this would naturally decrease the communication and information sharing with the product manager and product owner.

Furthermore, we felt that the general mood in the workshop did not necessarily support admitting that one does not know something. This is a very important issue to consider both inside the team and in future workshops - especially, since some team members reside in India, and cultural differences should be taken into account when creating a supportive atmosphere in a workshop.

We also encourage any future workshops to be held locally, and not in Skype. In the second workshop we noticed a clear decrease in the communication, and we are doubtful of how UX exercises could be successfully executed when the team is not in the same location. Due to the large distance between the team members, we would also consider holding two workshops - one in India and one in Finland.

The feelings of the interviewed persons regarding the workshops ranged from slightly positive to perhaps slightly negative. One developer stated that it is always good to hear of new thoughts and ideas, and felt that the workshops were not a waste of time but rather an incidental event. Another developer said that he had "in principle a positive feeling about the workshops", while the third developer felt slightly confused.

In regards to the future one developer mentioned how it could be beneficial to introduce the UX tools to all Napa teams at once to create a common way of software development, and also wished for more communication between teams. Two developers also wished for more hands-on exercises, which we do strongly agree with.

In regards to holding exercises in a distributed team, none of the developers had a clear cut answer. One contemplated holding UX workshops among only those team members, who are doing said feature or product, and another thought about location-centirc workshops, so that results of e.g. UX lab sessions would then be shared online.

One developer also stated that Napa should focus on forming a central UX team, as he felt that "we will not become UX professionals in a year or two". He was not sure whether the team could be formed of outsourced consultants or if it should be Napa's own team.

It was also suggested by one developer that the tools could be eased into the development process through the team's own demos by asking the team members how these UX matters are apparent in the demo.

Results of Introduced Methods

In the matrix below the observations made during the workshops and the comments from the team are gathered. Due to the lack of discussion and the tight schedule, there was not much to observe and analyse during the workshops. Thus, the most important observation was simply that there are challenges in the communication of the team.

Method Name	**Observations**	**Summary of the interview**
Personas	The lesson learned - that there is no shared understanding of who the user is - was valuable, but equally important was to notice that of the user was only between few team members.	The method was mentioned by three developers, one stating that he remembered it because it was an exercise where the team had to create something, and another remembered how the Indian developers had found it useful.
Nielsen's 10 Heuristics	No detailed observations could be made due to lack of discussion on the subject.	No comments in the interviews
Product Canvas	No detailed observations could be made due to lack of discussion on the subject.	No comments in the interviews
Crazy Eights	Due to lack of time the exercise could not be executed.	No comments in the interviews
Experiences of the Hull Team	The experiences shared by a Hull team member did spark some conversation, but only between a few Safety team members and the Hull team member.	The discussion was mentioned by one developer, who contemplated if it would be more beneficial to have two whole teams sit down together.
Asking the right questions	No detailed observations could be made due to lack of discussion on the subject	One developer mentioned the method in the interview, stating that checklists are good when they are short and easy to go through.

Table 4.1: Observations and summary of the interviews of the Safety team

4.3 Formation of the "UX Checklist" Method

After the workshops with the Safety team were completed, we began to further analyse the challenges in the team communication, and began to develop the "asking the right questions" concept further. We reviewed the challenges of the teams, uncovered in chapter 4, and summarized them as the following:

- Lack of understanding the big picture
- Challenges in communication between teams, inside a team, and between the developers and company management
- Pressure to push out features at the expense of doing user-centered work

To provide a solution to these issues, we introduced methods described in chapter 5.2.1., the First Workshop. However, after the workshop had been held, we felt that these kind of unrelated tools, disconnected from each other and from any process descriptions were not enough. We felt that the main challenge, in the everyday life of a developer, is the lack of understanding of the user. Questions they might ask were "Who are they? What do they do in their offices? Where are their offices? How do they work?", and so on. Knowing the answer to these questions would mean that the developer would likely also have wider knowledge of the business case behind a product or a feature - e.g. "the user designs ships at a ship yard, and also oversees some part of the construction, and must be able to quickly save and close his design work, and return to it easily. Thus, we should build a feature, that allows the user to do this."

We began to see the pattern of the basic information-gathering questions - who, what, why, how, when, where, also used in journalism - and based our method on those.

This formulated the first version of the UX Checklist method, "asking the right questions", which we introduced to the Safety team in the second workshop. However, after introducing the method, and not getting much feedback, we realized that again, as a lonely, separate method, without any

connection to the day-to-day life of the developers, the method would likely be quickly forgotten and stay unused.

As determined by Seffah and Metzker [2004] and mentioned in chapter 2, UCD and software development processes must become one, or UCD might not be properly integrated in the software development at all. Thus, we created the UX Checklist - a set of basic questions that each developer must be able to answer, and must answer, in every sprint review, retrospective, or other applicable event. Using this method should be written formally to any relevant document and enforced for at least some time, in order for the team to have time to adapt and truly evaluate the usefulness of the method in their respective teams.

To support this binding of the method and different phases of software development processes, we divided the method into two separate cases: questions to ask when building a completely new product, and questions to ask when updating an existing product.

Answering the questions in the Checklist would mean that the developer is knowledgeable of who the feature or item he is developing is developed for, and where it sits in the larger picture. The questions themselves are such that using other methods we introduced, such as Personas and Crazy Eights sketches, the developer or developers can gain good insight relatively quickly.

We also included in the Checklist:

- the act of sharing ideas and thoughts to ensure that at least two persons had thought about the solution and to make communication of ideas a norm,
- The sharing UI ideas to make sure that at least two people consider what the best UI might be,
- and, finally, Nielsen's 10 heuristics to ensure that the UI suggestion is not drawn without thought, but rather the developer can tell rationalize to the team why the UI he ended up with is the best.

A new feature	**An update to a feature**
Who... is the user of this feature?	Who... is the beneficiary of this update?
What... does he do with this feature?	What... will this update help the user to achieve?
How... does he use this feature?	How... Does this update make this feature better or the user's life easier?

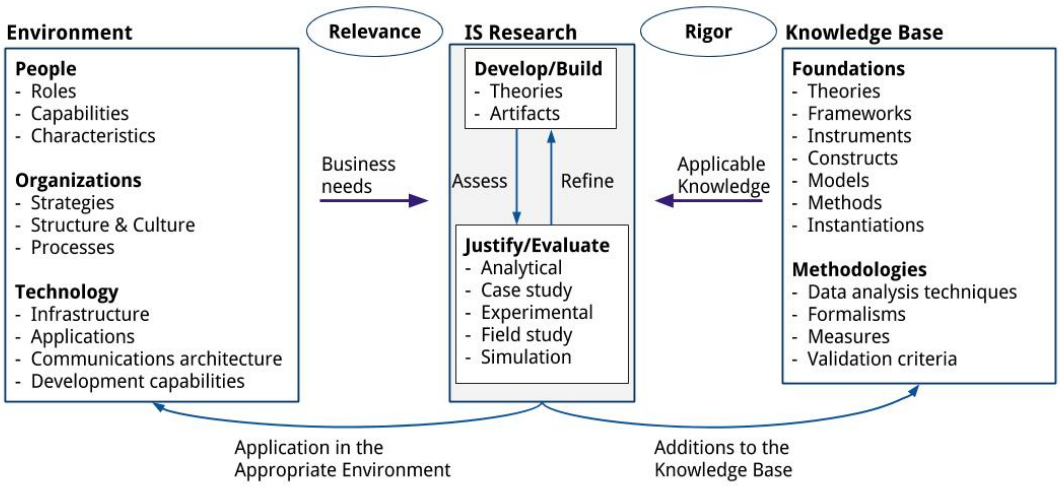

Figure 4.1: Information Systems Research Framework [Hevner et al., 2004]

Why... does he need this feature?	Why... is this update needed?
When developing... the process/UI, I shared my ideas with someone. The result was that... And the UI solution is good because...	When developing... I took the user into consideration by...

Table 4.2: The UX Checklist

The design and creation process of the UX Checklist is based on the the Information Systems Research Framework of Hevner et al. [2004], illustrated in 4.1. The Environment factors - people, organization and technology - were discovered in the interviews and the Knowledge Base is the literature review done in chapter two of this paper. The final step, assessing and refining, was done only partially and only with the VO team, due to scheduling difficulties. The results of introducing the UX Checklist are gone through in the next section.

4.4 Implementation and Results of workshops in the Voyage Optimization Team

With VO we had some scheduling difficulties, and managed to hold just one workshop near the end of the research period. However, it proved to be a more successful way of introducing UCD than holding multiple workshops over a long period of time. Similarly to chapter 5.2 regarding the Safety Team, here we first describe the implementation of the workshop, summarize the methods introduced, and finally provide a matrix of observations and evaluations of the introduced methods.

4.4.1 Implementation of the VO wokrshop

The agenda of the workshop was a follows:

1. What is UCD and why are we in this workshop
2. Experiences of the Hull-team
3. Understanding the user - Personas
4. Tools for UI design - Nielsen's 10 heuristics and Crazy Eights
5. Understanding the big picture - Product Canvas and UX Checklist.

The workshop was limited to two-and-a-half hours at most, with a coffee and bun break, as per Napa's Friday tradition, at the middle of it. The workshop consisted of a power-point show, hand-drawn prototypes, print-outs of the heuristics, and the real product of the VO team was used in a heuristics exercise.

The beginning of the workshop was relatively brief, with a focus on why UCD is useful to software development, and an emphasis on how the aim is not to make developers into UX specialists. The introduction was followed by a discussion with a Hull team member, who joined the workshop briefly to tell of the Hull team's experiences working with UCD. The discussion was more lively with the Hull Team member and the Voyage Optimization team, than it had been with the Safety team, with multiple team members taking part in the conversation and asking questions. This could be mostly due to team dynamics and personalities, but the medium of the discussion with the Safety team, Lync, could have also effected the discussion's activity.

After the discussion with the Hull team member we quickly moved to the UCD tools.

Personas

The first UCD tool presented was Personas [Holtzblatt et al., 2004], and after a brief explanation we began to create a user-persona. The exercise went fairly well, though there was discussion in the beginning on what kind of a persona should be created - for an existing product, or a future one; for a ship captain, or some other user. We suggest that in the future the Persona should be decided on beforehand with, e.g., the product owner of the team.

All team members took part in the discussion from the beginning, some more than others, but we felt that this was due to simply different personalities. Though the Persona was not completed, as is only natural when there is limited time, we consider it to have achieved its goal - spreading information, prompting conversation and introducing the tool to the developers.

Nielsen's 10

The second tool introduced was Nielsen's 10 heuristics [Nielsen, 2005]. For the presentation we had searched for websites and images of applications that portrayed good or bad UI design decisions, and reflected those against the heuristics. After going through the ten examples and their respective heuristics, we asked the team if there were any features in their current product that might be good for a heuristics exercise.

After some discussion the team chose an already-implemented feature, where a modern date-time picker had been implemented to replace an older one. The team's scrum master then went through the heuristics one by one, and thought out loud how the solution was compliant to the heuristics. While experiential learning (e.g. Kolb et al. [2001]) is an appropriate way of learning how to use a new tool such as Nielsen's 10 heuristics, in the future we recommend acquainting oneself to possible reviewable items and features. The feature that was gone through was possibly slightly too small, meaning that many of the arguments of why the solution was compliant to the heuristics were either repetitive or a heuristic was not applicable to the feature. On the other hand, it might be easier to evaluate only a small feature at first, and only after that start to assess larger entities. Be that as it may, one thing we absolutely do recommend is to get at least some kind of an understanding of the product and current backlog items, so that the feature to be evaluated can be chosen with knowledge of its applicability to the heuristics.

Crazy Eights

The second practice on UI design was the Crazy Eights of the Google Design Sprint [Google Product Design Sprint]. As we had understood from our initial interviews, the team members rarely shared their sketches on new interfaces and ideas. To combat shyness and awkwardness, we chose a completely unrelated theme - web shops. We gave all team members a free choice of the web shop's business area, and then started the clock. After five minutes, we asked them to choose one sketch, gave a few moments for consideration, and then gave three minutes for sketching a more detailed version of one of their ideas. This sketch would then be shared with the other team members.

After the five minutes were done, none of the team members had eight sketches on their paper, and after three minutes, most were not finished with their more detailed sketches. Despite not being able to finish the sketches, the exercise was successful - not only did it force all attendees to pick their brains for ideas, it also showed how many possible ways there are to implement even something as widely known as a webshop.

The detailed sketches we shared were very varying in regards to the UI solutions they depicted. We hope that after the exercise the general feeling is slightly more open towards sharing ideas and sketches, and maybe understanding how the first idea might not always be the best one, but rather might evolve into one after a few iterations.

Product Canvas

The third group of tools focused on the big picture of a product and integrating user-centered development into the agile software development process.

The Product Canvas of Pichler [2014] was gone through relatively quickly, as we felt that the team's challenges were more related to users and UI design rather than communicating and building the big picture of their product. Still, we view the Canvas' as a good tool for the team to use, as it brings visibility and helps to understand the big picture.

In the future, if the Product Canvas is needed, a meeting with the developers, PO and PM should be organized and the meeting should focus only on the Canvas. Depending on team and company culture, the Canvas could be filled so that all interested parties can participate and give input to the product, but in some companies this might not be customary. In this case, we recommend that the PM, or whomever is in charge of creating and updating the future plans of a product, fills in the Canvas in as much detail as possible, and then goes over it with the rest of the team. In both cases the Canvas should be put in some visible, central place where it is easily

accessible by all.

UX Checklist

The final tool introduced was the UX Checklist created by us. At this point the time was running slightly out, and we did not have time for an exercise, which would naturally have been preferable to simply introducing the tool on paper. In the future we suggest that the workshop facilitator works together with the team to choose appropriate team members and features to have the exercise on. The chosen team members would then answer the Checklist questions when applicable, and, if the team is supportive, the exercise could also be run on all developers. Furthermore, we highly encourage the team to implement the Checklist for at least a few sprints so that they get acquainted with the tool.

4.4.2 Results of the VO Workshop

The workshop was found to be of good quality by the interviewed developers, one stating that he preferred an even longer one to multiple shorter ones, and the other that there could have been even more information.

Responses regarding specific methods are gathered in the table below, and in addition to these one of the developers said that he had already begun considering how these methods could be brought into practice. He suspected that once they had been used in the real development process binding them into it would become easier. The other developer was of similar opinion, speculating that a having a structured way of using the methods in practice would be beneficial.

An improvement suggestion of deciding the subject of the Nielsen's 10 heuristics exercise before the workshop was made by one of the developers, as he had found it challenging to suddenly try and think of a good feature or product to use in the exercise. We agree with this suggestion completely. The other developer recommended considering how these methods could be brought to the development process directly before holding a workshop.

Results of Introduced Methods

Below are the observation results and the interview results regarding the different introduced methods.

Method Name	**Observations**	**Summary of the interview**

Personas	The team conversed actively of the persona to be created, and exchanged opinions about the most typical attributes and needs before settling on any. Conversation was made in good spirit and most of the team took part in the discussion. The Product Owner of the team was not among those most actively discussion the Persona.	One developer mentioned the Personas, stating that simple tools like Personas are good.
Nielsen's 10 Heuristics	The exercise was mostly monologue as one developer went over the heuristics. We do however feel that using an existing product or feature as something to evaluate with the heuristics is a good notion.	Both developers mentioned the tool, one stating that it was good to have hands-on exercises.

Crazy Eights	The team seemed clearly surprised on how little they could manage in the five minutes that were allocated to drawing. At the end of the exercise we felt that everyone did have fun and hope that the positive experience encourages the team to repeat it in the future.	Both developers mentioned sketching, one stating that it was good to have something concrete to do. He continued that it was beneficial that the exercise subject was not related to Napa, as people might be close up if they had to suddenly do a work related sketch. The other interviewed developer supported this, as he suspected most team members would already have an idea of how to create or renew a Napa product or feature. After stating this, he did wonder if by that logic it would be good to do the Crazy Eights for a Napa product, as it might make the participants question their original thoughts and ideas, when they had to make multiple sketches at once.
Product Canvas	No discussion	No mention in the interviews
UX Checklist	Though we did not have time for an exercise and thus no observations could be made, the method seemed to be well-received as we were asked to forward the method by e-mail by two team members.	Both developers mentioned the Checklist in the interviews, stating that it was good due to its simplicity. One developer went further on to describe how these kind of very simple and static methods are better than very flexible definitions. He gave an example of the Definition of Done, and how its ambiguity has caused it to be largely forgotten.

Conversation with Hull Team Member	Good discussion with questions and comments	Both developers found the conversation to be useful.

Table 4.3: Observations and summary of the interviews of the Safety team

Chapter 5

Discussion

In this chapter we first present a summary of the observations we made during the workshops and then detail the actions we recommend for Napa to take, and then discuss the limitations of the research and contemplate future work.

5.1 Research questions

In our research we aimed to answer the questions: "how can UCD be introduced and integrated into Scrum development in a company with very few or no UX professionals" and "what kind of UCD and UX tools would developers find valuable when developing software". We based our research on a literature review, collaborated with three teams, and held a total of three workshops.

Regarding how UCD and UX should be introduced to teams, we found that the following items were of importance to realize and consider:

1. One long workshop over multiple smaller ones.
2. Hands-on exercises of tools being introduced
3. During introductory and educational events, Lync workshops should be avoided.
4. Communication should be critically evaluated and, if deemed necessary, improved inside the teams, between teams, between departments and between organization levels.

When integrating UCD and agile software methodology together, we found the following four items critical to success:

1. The chosen methods and tools should bw written down to the process description and their use enforced for an appropriate time. These tools can then be evaluated and their use modified to better suit a team's needs

2. Share experiences and best practices of one team should be actively shared with other teams.

3. The integration should be started with small, simple, and unambiguous tools that are easily integrated into the software development process

4. Team spirit should be built so that it supports asking questions and learning new skills.

The first tool we encourage the teams to use is the UX Checklist, as it entails sketching, understanding what good usability is, collaboration with other team members, and understanding users and the use contexts of the product. However, even in the case that some other tool is experimented with, we strongly recommend it to be fully integrated into the development process for at least a few sprints.

5.2 Evaluation of the applicability of introduced Methods

In table 6.1 we evaluate the applicability of the introduced methods in regards to the needs recognized at Napa. Based on our observations and evaluations we strongly suggest researching the potential of the UX Checklist further, as it encompasses most of the other introduced methods, which we also feel were useful and provided valuable information to the developers. Depending on the project or case, not all of the methods introduced in this research need to be used, though we do encourage using them all to assure all-encompassing understanding of both the users and the big picture of the project or product.

Method Name	**Observations**

Personas	The use of Personas is encouraged very strongly, as it was found useful in both teams. During the workshops we observed how both teams discussed who the most typical user is, and in the interviews four of the five interviewed developers mentioned the method. As the Hull-team already has some Personas and has found them to be usable, utilizing them in the product development process could assist the teams in sharing knowledge and understanding the needs of the users better.
Nielsen's 10 Heuristics	While the discussion in neither group was very strong during the introduction of the 10 heuristics, we do feel that the tool should continue to be used. The comments of the method were neither negative nor overly positive, but the method was mentioned by two of the interviewed developers, which we see as a positive sign. Additionally, as per Curtis and Nielsen [1995], we believe that even minor usability evaluations and knowledge of the subject will make a difference in the overall usability of any product.
Product Canvas	The Product Canvas is a tool suited for understanding and communicating the larger and more complex view of where a product sits in the product family, why a product or feature is being developed, what goals the product or feature should reach and so on. While there was not much time to introduce the tool in any of the workshops we do feel that the Canvas could be a useful method that allows the team to better create a joint understanding of a product.
Crazy Eights	Although the method was only tried out in the VO workshop, the exercise generated such a positive atmosphere and demonstrated how quickly UI designs can be created that the method should be integrated into the development process.

UX Checklist	Based on the results of the interviews we also created the UX Checklist, consisting of the methods we introduced, to provide one, all-encompassing method that development teams could easily experiment with. Even though the method was demonstrated only in the VO workshop we encourage experimenting with it as it binds together all of the above methods.
Experiences of the Hull Team	Finally, while encouraging cross-team communication and sharing of experiences is in itself beneficial, we do suggest further evaluation of the method before using it. Though one VO developer found the discussion beneficial, which would encourage the use of the method, one Safety team member found the discussion difficult to follow. This to us indicates that for any conversation to be found useful and worth the time it takes, both participants should be invested in the discussion and potentially already have a basic knowledge of the subject. Therefore, we encourage Napa to further consider what kind of discussions would be most useful, and between whom they should be held.

Table 5.1: Summary of introduced methods

5.3 Recommendations

Based on the interviews and workshops we recommend five steps to be taken, with the three first being the most important:

1. Employment of a seasoned UX professional, who is also well-versed in change management and leadership.
2. Writing UX methods and milestones into the process description
3. Creation of Personas
4. Improved communication across teams regarding UX
5. Establishing a permanent UX team

Additionally, when integrating UCD and Agile software development methods it is always beneficial to follow established best practices such as those determined by Chamberlain et al. [2006].

.

5.3.1 Employment of an experienced UX professional

As there already is UX designing being done at one team, and the two other interviewd teams were interested in learning more of UX and UCD, Napa would greatly benefit from a seasoned UX Professional, who has experience on how UX can be integrated into an agile software organization. A UX Specialist is common role in many of the methods gone through in chapter two, and can be found in [Fox et al., 2008], [Google Product Design Sprint] and [Curtis and Nielsen, 1995], for example. These roles are not identical, and they should not be, as all companies and organizations are different and have different needs.

In our free-form discussion various employees at Napa described to us how some teams and persons are more open to change and shifting the focus on user-centered development, and how some teams and persons are less inclined to do so. An experienced change manager would help ease the transition, and they should be the same person as the UX manager, if possible.

The employment of a UX manager would alleviate the process of creating common guidelines and processes regarding UX and UCD to the whole company. The needs of teams seem to vary, with some teams needing more assistance in understanding who the users are and the users' processes, some requiring help in understanding the big picture Napa's products, and it is very likely other teams have different needs as well. A UX manager would be the person in charge of knowing what challenges teams face regarding UX, and managing resources on how to solve these challenges. They would also facilitate communication regarding UX between teams and together with teams integrate UX and UCD into the development process. Finally, as described in chapter two, Chamberlain et al. [2006] determine that the agile-UCD integration must exist within a cohesive project management framework. A seasoned UX manager would help in the creation and sustenance of such a framework.

5.3.2 Methods and Milestones

While writing UX Methods and Milestones into the process description would in effect force reluctant teams to take UX regularly into consideration, our work with the teams supports this way of integration.The idea of writing

the methods into the process description would also discounter the threat of separation of software development and UCD processes, as described by Seffah and Metzker [2004] in chapter two.

Writing the methods down was also supported in the final interviews with the developers, one Hull developer stating that when team members do not know what, for example, Scrum is, then it should be done to the letter in the beginning and only then start to modify it. He also recommended considering how UX and Scrum could be integrated right from the start, which was echoed by two VO developers, who contemplated and hoped for a structured way of integrating the two right from the beginning.

In the interviewed teams the feelings towards users and UX seemed to vary between "we know our users" and "it's a good idea but it's just not a good time right now to start considering UX". We believe that making UCD and UX a natural yet absolute part of the development process will help ease the transition in how teams see the two methodologies. In this the importance of a good UX/Change manager and management is again highlighted.

UX milestones that could be written into the process description vary from making user observations mandatory, e.g., every four weeks an observation session with an internal user is held, and at the end of a release cycle with an external user, to assuming that all developers can justify why they did the solution for the backlog item they implemented as they did. These justifications could include details such as being able to explain how the implementation supports the end-user at his work, how usability was taken into account, how the feature will support the whole product and why it is useful and so on. In the UX milestones some must-be-used tools could be defined, while leaving the possibility to determine other tools that could benefit the team to the team itself.

One tool we suggest experimenting with and including in the UCD process is the UX Checklist, as it provides an unambiguous, straightforward way of evaluating how much users were considered during the development of a sprint item. It also received positive feedback from four of the five developers who took part in the final interviews, with comments especially highlighting the its simplicity and unambiguity.

5.3.3 Personas

Furthermore, the creation of detailed and high-quality Personas representing at least one typical user per team, preferably more, would help all teams to understand what kind of users the Napa products have. Additionally, Personas help recognize and share knowledge of what kind of issues and

processes the users of Napa products have. Having quality Personas will both help developers to empathize with the end-users and allow the whole company to be better acquainted the processes real end-users use in their day-to-day work. All five developers also had positive feeling of the method, and the two Hull team members also mentioned the method, as the outsourced UX consultants had already created some Personas.

5.3.4 Improving communication

The fourth step of improving communication is not as vital for introduction of UCD and UX as the first three suggested steps, but improving cross-team communication would assist the process. This step is most important in the beginning of the introduction process, as there is currently minimal information sharing between the Hull team and other teams. Increasing communication is supported by the positive feedback from the discussion with the Hull team member from the workshops. Seffah and Metzker [2004] also determine, as described in chapter two, that key issues in integrating UCD and software development, among others, are lack of communication between teams.

One solution for how information sharing might be ensured could be holding one-on-one meetings between Hull and the other teams, where both teams' members could simply discuss their experiences, as suggested by one of the interviewed developers. However, methods for sharing information were not at the focus of this research and we leave the discussion for better information sharing channels and methods to the company itself and later research.

5.3.5 UX Team

Finally, one of the current difficulties within the Hull team is that the outsourced UX consultants do not work at Napa every day, for the whole day. This creates bottleneck situations, as the development team must still create something, while the consultants are not at Napa. Though there is one permanent UX-educated team member employed by Napa in the team, it is not enough to keep UX up-to-date with the development's implementation schedule. To help ease the bottlenecks we propose a UX team for Napa, to work with all teams, as per Ungar and White [2008] and [Sy, 2007], for example. The composition and size of the team would depend on what the needs across the different teams are - if there is continuously much need for visual design, then more visual designers should be employed. If UX focus is decided to be the responsibility of the teams themselves, and, for example,

they are to organize their own observation sessions, then less UX designers might be needed.

Again, at the beginning of these changes there is need for more UX designers, as the creation of Personas requires much observation and, in Napa's case, travel. To do multiple Personas, possibly from multiple countries and relatively quickly, multiple UX designers would be needed. A compromise for hiring personnel temporarily could be to, for example, use an international UX consultancy or a global UX network such as UX alliance. However, for the purpose of introducing and integrating UX and UCD into the development process, the employment of a UX manager is more important than creating a UX team.

5.3.6 Excluded methods

In companies and teams similar to Napa, of the methods summarized in table 4.2 we thus leave out the following:

- One Step Ahead [Sy, 2007]
- Discount Usability [Curtis and Nielsen, 1995]
- Design Studio [Ungar and White, 2008]

These tools and methods are not suitable to Napa and other teams and companies with little to no UCD support due to their demands for experienced UX and UCD designers.

The Rapid Contextual Design [Beyer et al., 2004] and the Specialist, Generalist, Hybrid method of Fox et al. [2008] were excluded because even if they are not meant strictly for teams with UCD professionals, they do require at least informal knowledge of all things related to UCD or otherwise require the developers to self-study, making their integration much more complex than simpler tools.

Finally, the UScrum of [Singh, 2008] is not recommended to Napa due to it requiring a very profound change in the work process by demanding a UX-centered product owner. As Napa's teams already have a product owner and a product manager it is better to focus on enabling the team members to take responsibility of user experience and user-centered development instead of outsourcing it to one new product owner.

5.4 Limitations and weaknesses of the research

One of the limitations of this study is that the workshops were not reproduced, and all workshops were different.

The first workshop with the Safety team was held locally with the Indian team members present, and only the Personas were used as an exercise. Nielsen's 10 heuristics were gone through but not used as an exercise. In the second workshop with the Safety team the setting was fundamentally different, as the workshop was held in Lync. Personas and Nielsen's heuristics were gone over in Lync, and the method of "Asking the right questions" was introduced.

With the Voyage Optimization team only one workshop was held, and in that Personas were used as an exercise as well as Nielsen's 10 heuristics and the Crazy Eights. The method "Asking the right questions" was also introduced as a more complete tool, the UX Checklist.

Thus, even if we can form recommendations based on these three different workshops, more research and repetitions of the workshops would need to be completed to better evaluate the applicabilities of the suggested methods.

The interviewed developers were also all Finnish and spoke Finnish in the interview, even though both the Safety and VO team have members from other countries. This might have limited the comprehensiveness of this research, as there might be cultural aspects that this research did not take into account.

The workshop-method was used for facilitation in this research was chosen simply due to its nature of bringing the stakeholders into the same room and encouraging conversation. More research is thus needed into the hands-on approach of how to introduce new tools and methods to the teams. The researchers were also not part of either team and did not share a room with either of the team, which prevented full immersion in the teams' cultures.

5.5 Future work

In future work, the research period should be long enough that changes can be implemented to the process description and see if and how teams actually begin to apply UCD methods. It would also be beneficial that the researchers were to be included more fully in the team - sitting in the same room, taking part in every daily scrum and so on.

It would also be interesting to see if the method we created, the UX Checklist, would prove to be useful in other teams as well, especially in the early phases of integration.

Finally, in any future work, it would additionally be interesting to see if and how a fully utilized UX championship-team and an experienced UX manager would help integrate software development and user-centered development more swiftly.

Chapter 6

Conclusions

As we describe in chapter one, there does not seem to be much research on how UCD can be introduced and integrated in companies that have little to none in-house UCD expertise. There also appears to be a lack of research focusing on what kind of tools developers could find valuable when looking to create software in a more user-centered way, and an overall challenge of how UX research and industrial UX development currently focus on different issues, as recognized by Vaananen-Vainio-Mattila et al. [2008]. The significance of this research in the UCD field is that it focuses on two teams just encountering these issues, and has attempted to recognize how teams with no UX personnel could introduce and integrate user-centered design processes and methods into software development, and especially the Scrum process.

The Hull-team was experiencing very similar issues to Sy [2007] in regards to how the user experience tasks and the coding of the product could be done in a convenient schedule. The research by Fox et al. [2008] produced a development process very similar to Sy's, but apart from these two papers we encountered no other research describing how UCD and Scrum have been integrated in practice, successfully or not. Despite the challenges of the scheduling, both interviewed developers felt positive of the integration and recognized that the introduction of user-centered development tools had been beneficial. This lack of research into how different kind of teams could begin to integrate UCD and software develompent could easily mean that each team and company seeking to integrate the twp must re-invent the wheel instead of taking an already-proven solution and developing a process suitable for them based on that. Furthermore, various team and company compositions should be investigated more - not only can there be variability in team culture and readiness for UCD, but also in how big a central UX team the company has, or is there one at all, or do all teams have one UX professional, and so on.

The research also evaluated the applicability and usefulness of five tools

the developers could benefit from - Personas, the Product Canvas, the Crazy Eights, Nielsen's 10 Heuristics, and the UX Checklist, which was created during the research process. All of the evaluated tools were such that the developers could use without the help of a UX professional, which again supports the goal of integrating UCD into a team with no UX professional. Only the research of Fox et al. [2008] considered software development teams with no UCD professionals, but the research did not evaluate what kind of tools such teams found beneficial. As we were not able to evaluate how the teams would have taken the tools into use, and if they would have found them beneficial and easy to use in practice, more research is needed into the suitability of various UCD tools in teams without UCD professionals.

However, the positive comments in the final interviews support our assumption that the UX Checklist could be a beneficial tool for the developers to use during the integration process, as the feedback revealed that small, simple tools are preferred. As we did not find any such tools, created specifically for developers, during our research, implicates there is a need for tools of this kind to be created.

Bibliography

Abras, C., Maloney-Krichmar, D., and Preece, J. (2004). User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, 37(4):445–456.

Beck, K. (1999). Embracing change with extreme programming. *IEEE Computer*, 32(10):70–77.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001a). Manifesto for agile software development. Accessed: 2015-08-22.

Beck, K., Beedle, M., Van Bennekum, A., 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., and Thomas, D. (2001b). Agile manifesto. `https://www.agilealliance.org/agile101/the-agile-manifesto/`. Accessed: 2015-08-22.

Bevan, N. (2009). What is the difference between the purpose of usability and user experience evaluation methods. In *Proceedings of the Workshop UXEM*, volume 9.

Beyer, H. R., Holtzblatt, K., and Baker, L. (2004). An agile customer-centered method: Rapid contextual design. In Zannier, C., Erdogmus, H., and Lindstrom, L., editors, *Extreme Programming and Agile Methods - XP/Agile Universe 2004, 4th Conference on Extreme Programming and Agile Methods, Calgary, Canada, August 15-18, 2004, Proceedings*, volume 3134 of *Lecture Notes in Computer Science*, pages 50–59. Springer.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21(5):61–72.

Boehm, B. W. (2006). A view of 20th and 21st century software engineering. In Osterweil, L. J., Rombach, H. D., and Soffa, M. L., editors, *28th International Conference on Software Engineering (ICSE 2006), Shanghai, China, May 20-28, 2006*, pages 12–29. ACM.

Brooke, J. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7.

Budwig, M., Jeong, S., and Kelkar, K. (2009). When user experience met agile: a case study. In Jr., D. R. O., Arthur, R. B., Hinckley, K., Morris, M. R., Hudson, S. E., and Greenberg, S., editors, *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Extended Abstracts Volume, Boston, MA, USA, April 4-9, 2009*, pages 3075–3084. ACM.

Chamberlain, S., Sharp, H., and Maiden, N. A. M. (2006). Towards a framework for integrating agile development and user-centred design. In Abrahamsson, P., Marchesi, M., and Succi, G., editors, *Extreme Programming and Agile Processes in Software Engineering, 7th International Conference, XP 2006, Oulu, Finland, June 17-22, 2006, Proceedings*, volume 4044 of *Lecture Notes in Computer Science*, pages 143–153. Springer.

Chapman, L. and Plewes, S. (2014). *A UX Maturity Model: Effective Introduction of UX into Organizations*, pages 12–22. Springer International Publishing, Cham.

Cooper, A., Reimann, R., Cronin, D., and Noessel, C. (2014). *About Face: The Essentials of Interaction Design*. Wiley.

Curtis, B. and Nielsen, J. (1995). Applying discount usability engineering. *IEEE Software*, 12(1):98–100.

Düchting, M., Zimmermann, D., and Nebe, K. (2007). Incorporating user centered requirement engineering into agile software development. In Jacko, J. A., editor, *Human-Computer Interaction. Interaction Design and Usability, 12th International Conference, HCI International 2007, Beijing, China, July 22-27, 2007, Proceedings, Part I*, volume 4550 of *Lecture Notes in Computer Science*, pages 58–67. Springer.

Elliott, R. and Jankel-elliott, N. (2003). Using ethnography in strategic consumer research.

Fox, D., Sillito, J., and Maurer, F. (2008). Agile methods and user-centered design: How these two methodologies are being successfully integrated in industry. In Melnik et al. [2008], pages 63–72.

Google Product Design Sprint. Google product design sprint. `http://www.gv.com/lib/the-product-design-sprint-settingthestage`. Accessed: 2015-06-07.

Halskov, K. and Dalsgård, P. (2006). Inspiration card workshops. In *Proceedings of the Conference on Designing Interactive Systems,University Park, PA, USA, June 26-28, 2006*, pages 2–11.

Heinbokel, T., Sonnentag, S., Frese, M., Stolte, W., and Brodbeck, F. C. (1996). Don't underestimate the problems of user centredness in software development projectsthere are many! *Behaviour & IT*, 15(4):226–236.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1):75–105.

Holtzblatt, K., Wendell, J. B., and Wood, S. (2004). *Rapid contextual design: a how-to guide to key techniques for user-centered design.* Elsevier.

Holzinger, A. (2005). Usability engineering methods for software developers. *Communications of the ACM*, 48(1):71–74.

International Organization for Standardization (2010). Ergonomics of human-system interaction – part 210: Human-centred design for interactive systems. `https://www.iso.org/obp/ui/#iso:std:iso:9241:-210:ed-1:v1:en`. Accessed: 2015-09-04.

Jokela, T., Iivari, N., Matero, J., and Karukka, M. (2003). The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11. In de Souza, C. S., Sánchez, J. A., Barbosa, S. D. J., and Gonzalez, C., editors, *Latin American Conference on Human-Computer Interaction, CLIHC'03, Rio de Janeiro, Brazil, November 17-20, 2003*, pages 53–60. ACM.

Kähkönen, T. (2004). Agile methods for large organizations - building communities of practice. In *2004 Agile Development Conference (ADC 2004), 22-26 June 2004, Salt Lake City, UT, USA*, pages 2–11.

Katz-Haas, R. (1998). User-centered design and web development. *Usability Interface*, 5(1):12–13.

Kolb, D. A., Boyatzis, R. E., Mainemelis, C., et al. (2001). Experiential learning theory: Previous research and new directions. *Perspectives on thinking, learning, and cognitive styles*, 1:227–247.

Law, E. L., Roto, V., Hassenzahl, M., Vermeeren, A. P. O. S., and Kort, J. (2009). Understanding, scoping and defining user experience: a survey approach. In Jr., D. R. O., Arthur, R. B., Hinckley, K., Morris, M. R., Hudson, S. E., and Greenberg, S., editors, *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009*, pages 719–728. ACM.

Liikkanen, L. A., Kilpiö, H., Svan, L., and Hiltunen, M. (2014). Lean ux: The next generation of user-centered agile development? In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*, NordiCHI '14, pages 1095–1100, New York, NY, USA. ACM.

Mao, J., Vredenburg, K., Smith, P. W., and Carey, T. (2005). The state of user-centered design practice. *Commun. ACM*, 48(3):105–109.

Mayhew, D. J. (1999). The usability engineering lifecycle. In Atwood, M. E., editor, *CHI '99 Extended Abstracts on Human Factors in Computing Systems, CHI Extended Abstracts '99, Pittsburgh, Pennsylvania, USA, May 15-20, 1999*, pages 147–148. ACM.

McInerney, P. and Maurer, F. (2005). UCD in agile projects: dream team or odd couple? *Interactions*, 12(6):19–23.

Melnik, G., Kruchten, P., and Poppendieck, M., editors (2008). *Agile Development Conference, AGILE 2008, Toronto, Canda, 4-8 August 2008*. IEEE Computer Society.

Nielsen, J. (1994). Guerrilla hci: Using discount usability engineering to penetrate the intimidation barrier. *Cost-justifying usability*, pages 245–272.

Nielsen, J. (2005). Ten usability heuristics. `http://www.nngroup.com/articles/ten-usability-heuristics/`. Accessed: 19.12.2014.

Nielsen, J. (2008). Nielsen norman group: The first decade. `https://www.nngroup.com/articles/nielsen-norman-group-first-decade/`. Accessed: 20.4.2016.

Pichler, R. (2014). Product canvas. `http://www.romanpichler.com/wp-content/uploads/2014/06/UX-and-Scrum.pdf`. Accessed: 2015-09-06.

Pruitt, J. and Grudin, J. (2003). Personas: Practice and theory. In *Proceedings of the 2003 Conference on Designing for User Experiences*, DUX '03, pages 1–15, New York, NY, USA. ACM.

Scrum Guide. Scrum guide. `http://www.scrumguides.org/`. Accessed: 2015-09-05.

Seffah, A. and Metzker, E. (2004). The obstacles and myths of usability and software engineering. *Commun. ACM*, 47(12):71–76.

Singh, M. (2008). U-SCRUM: an agile methodology for promoting usability. In Melnik et al. [2008], pages 555–560.

Sohaib, O. and Khan, K. (2010). Integrating usability engineering and agile software development: A literature review. In *Computer design and applications (ICCDA), 2010 international conference on*, volume 2, pages V2–32. IEEE.

Sy, D. (2007). Adapting usability investigations for agile user-centered design. *Journal of usability Studies*, 2(3):112–132.

Ungar, J. and White, J. (2008). Agile user centered design: enter the design studio - a case study. In Czerwinski, M., Lund, A. M., and Tan, D. S., editors, *Extended Abstracts Proceedings of the 2008 Conference on Human Factors in Computing Systems, CHI 2008, Florence, Italy, April 5-10, 2008*, pages 2167–2178. ACM.

UX Definitions. Ux definitions. `http://www.allaboutux.org/ux-definitions`. Accessed: 2015-09-05. The information on Allaboutux.org is collected from the UX community and is shared and maintained by volunteers: Roto, V., Lee, M., Pihkala, K., Castor, B., Vermeeren, A., Law, E., Vaananen-Vainio-Mattila, K., Hoonhout, J., and Obrist, M.

Vaananen-Vainio-Mattila, K., Roto, V., and Hassenzahl, M. (2008). Towards practical user experience evaluation methods.

Wichrowski, M., Korzinek, D., and Szklanny, K. (2015). Google glass development in practice: UX design sprint workshops. In *Proceedings of the Mulitimedia, Interaction, Design and Innnovation, MIDI 2015, Warsaw, Poland, June 29-30, 2015*, pages 11:1–11:12.

Appendix A

Appendix A

Semi-structured interview layout of the first phase interviews.

1. Please tell in your own words who you are, what you do as work, and which team you lead.
2. Please tell of your team and your product
   (a) Are all team members developers, or do they have specific skillsets or expertise?
3. What kind of stakeholders do you work with? At which points in the development/release cycle?
4. A new release is beginning - please tell us what kind of processes you use, what kind of schedule you have, and what kind of duties you have
   (a) Who decides what to do in a sprint/release?
   (b) How do you develope and communicate - pair programming, do you ask for help, etc.?
5. How does the team communicate and share information?
6. How do you manage the big picture and communicate with various stakeholders?
   (a) Definition of done, visuality, shared folders?
   (b) Do you know what other members of the team are developing?
   (c) Do you have a common ?goal??
7. How is requirements elicitation done (with different stakeholders)?

8. How do you manage and communicate the different requirements of various stakeholders?
9. Management of change - how are change requirements from various stakeholders recovered, managed and communicated to other applicable stakeholders?
10. Do you ever find out about issues in demos or retros that would have affected your work, if you had known of them earlier?
11. Please tell in your own words how the end user or the customer is part of the development process
12. How are the needs of the end user and/or customer recovered? At what point is this done?
13. Miten asiakkaan tarpeet ja/tai loppukäyttäjän tarpeet selvitetään? Missä vaiheessa?
14. Who (and how) is responsible that the final product answers to the end-user?s needs?
15. Who represents the end-user and/or the customer?
    (a) When an issue occurs, who is asked what the user would want, or does the developer in charge of the issue decide?
16. Please describe a typical user
17. At which point is feedback about the product typically gathered from the user and/or customer?
    (a) How is the feedback attempted to be gathered? At which point in the development process?
    (b) Do you use prototyping?
18. Do you have a good idea of what product you are developing, who you are developing it for, and why you are developing i?
19. From whom does the change request regarding UX/UI usually come from?
    (a) How do you attempt to gather the change requests? At which point in the development process?
    (b) Do you use prototyping?

20. How and when is the product tested?
    (a) By whom?
21. How is the product defined as ?done?? Who makes the decision?
22. Do you often have to change the product based on the feedback from the users or stakeholders?
    (a) Where do you think the need to change the product arises from? Is the requirements elicitation lacking, are the needs of the stakeholders not taken into account or interpreted correctly, or are there challenges in the communication?
    (b) Is the ?big picture? managed?
    (c) How is the communication inside the team between the developers?
    (d) Is it common that plans for a release/sprint have to be changed? Can you say a reason for this.
23. At what point do you feel the help of users would be most beneficial? What kind of help could they offer?
24. What kind of tools do you think would benefit the team/What kind of tools would you hope for? Do you have motivation for learning of UCD?
25. Is there more pressure to release new features than there is to do bugfixes?
26. Have you had observation of users in your team?
27. Anything else to add?

Appendix B

Appendix B

Semi-structured interview layout of the second phase interviews.

1. How do you feel now that the workshops have been held?
2. Did anything stick with you? Was it better to have the introductions during a long time/short time?
3. What did you feel was most useful?
4. Was there anything ?unnecessary??
5. Was there anything missing?
6. Was there anything frustrating in the workshops?
7. How could the workshop have been organized better?
8. How could the discussion have been encouraged?
9. What were the reasons for the challenges (e.g. lack of conversation)?
10. Do you feel that the workshops via Lync could be successful? How do you think workshops in a distributed team should be organized?
11. Do you feel that the product owner, product manager, and developers should be in the same workshop?
12. How do you now feel about UX/UCD?
13. Did your view on UX and UCD change during the research?
14. Were the introduced tools useful?

15. Do you feel that there is a need for tools like the introduced ones?
16. Have you taken any tools into use? How could they have been taken into use?
17. Were discussions with the Hull team member beneficial?