User-Centered Design in Agile software development for in-house enterprise tools

Name: Samuel Farebo
E-mail: samuel.farebo@gmail.com

Supervisor at Umeå University
Ulrik Söderström

Supervisor at Com Hem AB
Mikael Rask
Abstract

The Agile software development model is driven by "learning by doing" and rejects Big Design Up Front (BDUF) for that reason. User-Centered Design (UCD) on the other hand requires a more holistic view to be able to create a usable user interface and in the end create a good user experience [1, 2].

Finding a balance between the incremental development and the need for a more comprehensive view of the user interface is therefore the key to usability in Agile software development. The objective of this master thesis was to construct a framework on how to combine UCD and Agile development in general, and specifically for the web based tool, called Alo, at the IS/IT department of Com Hem AB, Sweden.

The results of this thesis was that the process of integrating User-Centered Design in Agile software development first of all needs a familiar starting point for both usability experts and developers. This can be achieved with what Desirée Sy describes as “Cycle Zero” [3], to let usability experts perform initial research ahead of implementation. Designing one sprint ahead should later converge to a more synchronized process where requirements and sketches of the interface are put together, with the help of developers, just in time for the implementation. This does not only prevents waste in the form of documentation and miscommunication associated with hand-offs, but also makes the implementation more purposeful and fun for developers. Secondly, build prototypes early in the process to create a holistic vision of the finished product and to test concepts in usability tests early. Thirdly, create shared understanding (within the development team as well as with outside stakeholders) of user needs by involving the entire team in usability testing.

Critical to the success of all the above is that all outside stakeholders understands the Agile process and respects that the team is a self-organizing unit [4] that solves problems within a set of given boundaries, rather than a code factory that feeds on specification documents.

Keywords

User-Centered Design, Agile, interaction design, usability testing.
# User-Centered Design in Agile development

## Contents

1. **Introduction**  
   1.1 Towards an Agile and User-Centered Design Process at Com Hem  

2. **Objective**  

3. **Definitions**  

4. **Background**  
   4.1 About Com Hem AB  
   4.2 About the Alo Project  
   4.3 The Goal With Alo  
   4.4 How Alo Came to be  

5. **Related Work**  
   5.1 Constantine & Lockwood 2002  
   5.2 Stefan Blomkvist 2005  
   5.3 Desirée Sy 2007  
   5.4 Salah et al. 2014  

6. **Method**  
   6.1 Literature Review  
   6.2 Retrospective Analysis of the Alo Development Process  
      6.2.1 Defining Requirements for Alo  
      6.2.2 Sketching on Paper  
      6.2.3 Creating Digital Wireframes  
      6.2.4 Usability Testing  
      6.2.5 Using high-fidelity prototypes  
      6.2.6 Usability Testing with Expert Users  
      6.2.7 Using a Pilot Group  
      6.2.8 Surveys With Customer Service Agents  
      6.2.9 From Scrum to Kanban with XP  
      6.2.10 Roles in the Alo Development Team  
   6.3 Survey With Developers  
   6.4 Involving Developers in Usability Testing  
   6.5 Limitations  

7. **Agile Development Processes**  
   7.1 Lightweight Development Processes  
      7.1.1 Scrum, XP and Kanban  
   7.2 User Stories  

8. **User-Centered Design**  
   8.1 Making Sure it’s Usable by Testing it  
   8.2 Weaknesses With Qualitative Usability Tests  

9. **Results**  
   9.1 Result from Survey with Developers  
   9.2 Results from Surveys with Customer Service Agents  
   9.3 Usability Testing High-Functionality Interfaces like Alo  
   9.4 Problems in the Alo Development  
   9.5 Successes in the Alo Development  
   9.6 Remote Usability Testing  

10. **Future Work**  

---

Samuel Farebo, samuel.farebo@gmail.com  
October 27, 2015
1 Introduction

"A system is not usable unless it is both released and put to use."
- Stefan Blomkvist [5]

As the world has become more and more driven by information technology, almost every company has in some part become a software or digital service manufacturer. Many of the traditional companies have moved to the digital realm or have been replaced by software driven competitors. For example, the American video rental company Blockbuster, had at its peak in 2004 over 60 000 employees [6] and more than 9 000 stores [7]. Blockbuster filed for bankruptcy in September 2010 [8]. Blockbuster had been out-rivaled by companies like Netflix [9] that launched its streaming service in 2007 [10], making it faster and more convenient for customers to rent movies. Amazon, the online retailer, had become the 9th largest reseller in the United States by 2014 [11], competing with giants like Walmart and Target. Amazon was also the first retailer on the top ten list to exclusively sell via an online store [11]. As digital services has grown in importance for companies to stay ahead of competition, so has the importance of delivering those services at a faster pace and giving the user a better experience [12].

Although incremental software development methods are nothing new, dating back as far as the mid 1950s [13], their relevance has increased by enabling companies to be innovative in order to stay ahead of competition [14, 15]. One of the key benefits of the Agile framework for software development is the concept of delivering functionality to the user to evaluate if what you as a developer have built is of any value to the person using the product, thereby reducing time-to-market [12].

As Steve Jobs noted at Apple’s Worldwide Developers Conference (WWDC) in the San Jose 1997:

"You’ve got to start with the customer experience and work backwards to the technology. [16, 52:15]"*

With the ever growing use of the Internet, customer and user experience has become an important factor in the success of services [17]. The goal with User-Centered Design (UCD) is exactly what Steve Jobs expressed in the above quote: Put the user at the center of the design process [18]. While Agile focuses on the process of developing, and especially activities surrounding programming, the main focus of User-Centered Design is the usability of the product and the evaluation of it [5]. Both Agile and UCD share traits (i.e. incremental improvement and iterating processes) but the contrast in thinking can be hard to reconcile in practice since Agile main focus is fast delivery, for shortening the feedback loop and deliver value early 1, and UCD on long term vision of the user experience. Building a coherent interface requires research up front to capture the users need and context of usage [20]. Acquiring knowledge about the users requires thinking beyond just the currently developed feature or ongoing sprint.

---

1"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software." [19]
1.1 Towards an Agile and User-Centered Design Process at Com Hem

While replacing outdated internal systems (a.k.a "legacy systems"), Com Hem demanded shorter delivery cycles and continuous delivery instead of waterfall process with scheduled monthly releases, that until that point had been the standard at the company [21].

Com Hem also realized that usability testing and user involvement in the development was extremely important for the success when introducing new tools to its employees. This knowledge came from the previous implementation of a commercial off-the-shelf (COTS) CRM system in 2009, that eventually resulted in longer average handling time (AHT) per call from customers permanently, not just initially at the introduction. Increased handling time means not only that customers have to spend more time on the phone getting help, but also spending more time in queue, before an agent can answer their call. This leads to frustration and irritation not only among the customers but also effecting the Customer Service Agents (Agent, in short) taking the calls.

With those insights, Com Hem chose to let an outside team of designers perform a feasibility study, and observe how the Customer Service Agents worked and interview key specialists within the organisation, to understand what type of problems they were helping customers with. Com Hem asked the team to disregard any technical limitations, to promote new and innovative solutions.

The study resulted in a Proof of Concept (PoC) prototype handling some of the most common usage scenarios.

The PoC was well received by the management team and Com Hem decided to form a team with developers and usability designers, working by the principles of Lean, Agile and User-Centered Design, to build a new tool for its Customer Service Organisation.

2 Objective

The main objective was to find a way to fit User-Centered Design (UCD) with Agile software development within the Alo project to ensure usability of the Alo application, while maintaining delivery speed and minimizing waste for the development team. A literature review was performed to find ways on how to incorporate Agile and UCD in the development process at the IS/IT department at Com Hem.

The secondary objective was to spread knowledge, within the IS/IT department at Com Hem, about the integration of agile development and UCD in the Alo project in general, and usability testing techniques specifically.

3 Definitions

- IS/IT - The Information system and information technology department at Com Hem. The IT personell works with ensuring that the services
Hem provides are delivered to the customer and that servers are running as they should. The IS personell takes care of developing and maintaining the software tools, like the billig system or the external website. The Alo development team is a part of the latter.

- **Average Handling Time (AHT)** - The time it takes to answer a customers call (or e-mail), help the customer and document what was done by creating a customer ticket. The shorter, the better for both the customer and Com Hem.

- **Usability** - according to the ISO standard 9241-11:

  "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. [22, p. 186]"

- **Usability testing** - to have a representative user perform realistic tasks on a prototype of the system being developed.

  "During a usability test, one or more observers watch one or more participants perform specified tasks with the product in a specified test environment...[23, p. 1269]"

In the case of Alo, usability tests were mainly performed on a computer with screen recording software.

- **Low Fidelity Prototype**

  Low fidelity (lo-fi) prototypes are mockups of the system being built, made in paper or digital form. The lo-fi prototype only conveys the basics of the interface, meaning functionality and rough layout. Lo-fi prototypes does not incorporate the finished design/coloring/graphics [23, p. 1345].

  - **Paper prototype** - A paper based version of a system or interface, used for designing, brainstorming and usability testing. As Carolyn Snyder puts it in her book Paper prototyping [24, p. 4]: "Paper prototyping is a variation of usability testing where representative users perform realistic tasks by interacting with a paper version of the interface that is manipulated by a person 'playing computer,' who doesn’t explain how the interface is intended to work."

  - **Wireframe** - a drawing or illustration of an interface that focuses on where content and functionality is located and what space is allocated for it. A wireframe typically lacks design elements such as graphics, color and typography. [25]. In the case of this thesis, the wireframes used were varying in range from lo-fi to hi-fi printed on paper or as static images.

- **High fidelity (hi-fi) prototypes** is supposed to be as close to the finished product design as possible with much more detail than the lo-fi prototypes [26].

  - **Interactive wireframes** - wireframes with added interactivity (i.e fake logic) and design.

  - **Coded Prototype** - "...exist in the native environment (the browser,
the OS, on the device, etc.) and make use of all of the expected interactive elements. Buttons, drop-down menus, and form fields all function as the user would expect them to." [26, p. 178]

- **Agent** - Short name for Customer Service Agent. The person taking support calls or e-mail from customers.

- **Alo** - An in house built customer relationship management (CRM) tool as well as a customer ticket interface, custom built for the customer service department at Com Hem.

- **MVP** - Minimum Viable Product. An expression, popularized by Eric Ries [27], describing how to test if what you are building (computer code, a service et cetera) will bring your users/customers the expected value, with minimum effort.

- **MLP** - Minimum Lovable Product. MLP is sometimes used instead of MVP to avoid the negatively charged word "viable" [28]. In the Alo project, an MLP was the next step in the development process. By learning from the feedback on the MVP functionality, an MLP was developed. One that the user would like to use.

- **LP** - Lovable Product. A definition that the program manager of the Alo development team coined, to describe a functionality that is as close to perfection as possible, by learning from user feedback on the MLP. It could also be an extra feature or polishing a feature that the user may not explicitly require to perform their duties, but will indeed love when using it.

- **PO** - Product owner. The person (or in the case of the Alo development, two persons), responsible for placing first draft of user stories in the product backlog and prioritizing what the development team should build next. In Alo, the PO:s were also business experts from the customer service department.

- **User story**. Stories can be seen as requirement from the eyes of the user, describing what the user would like to do and the desired outcome after said action. User stories are written in a technique agnostic way that every stakeholder can understand. See section: section 7.2 below, for more details.

- **Use case**. In contrast to a user story, use cases describe actors’ (i.e. a customer, a buyer, a computer system and so on) interactions, step by step. Use cases usually describe the best case scenario. Use cases can be technical and does not take the users desired outcome in account.

4 **Background**

This master thesis has been written during the spring of 2015 at the IS/IT department at Com Hem AB in Stockholm, Sweden.
4.1 About Com Hem AB

Com Hem AB is one of the largest supplier of broadband internet, cable tv and telephony in Sweden with more than 875 000 customers and nearly 1.9 million connected households [29, p. 8]. The company was founded in 1983 under the name "Svenska Kabel-TV AB" before becoming Com Hem in 1999 [30]. The customer service department of Com Hem is located in: Sundsvall, Härnösand and Örnsköldsvik, and employs about 800 people (including part time resources) [29, p. 10] and each customer service agent handles about 75 calls a day [29, p. 25] resulting of several thousands of interactions with Alo each day. The customer service headquarters is located in Härnösand, a town with about 25000 inhabitants, situated slightly north of the geographical center of Sweden.

The majority of the usability tests described in this master thesis (19 in total) has been performed with the help of the agents from the Härnösand and Örnsköldsvik offices.

4.2 About the Alo Project

This master thesis is a part of the development of Alo\(^2\), an in-house built customer relationship management (CRM) tool as well as a customer ticket interface, custom built for Com Hem. Alo is a web based tool, built with Java (back-end), Javascript and HTML (front-end).

The users of Alo are the customer service agents (or agents, in short) taking calls and answering e-mails from the end customers of Com Hem’s tv-, broadband- and telephony services.

4.3 The Goal With Alo

Com Hem’s goal with Alo:

1. Replace the interface of several legacy systems thereby cutting costs in development/maintenance and reducing the number of tools that the agents have to interact with on a daily basis.

2. Cutting down the average handling time of every call, by making the tools more usable and fit the agents’ needs better.

3. Implementing sales tools to increase revenue and customer satisfaction.

4. Being the most loved internal tool with the best user experience for the customer service agents.

4.4 How Alo Came to be

The Alo project started as a proof of concept (PoC) in the March 2012 with a two week research and prototyping period. The PoC prototype was produced

\(^2\)Alo is Latin and means: nourish, cherish, support
with the users (i.e. the agents) in mind and actively without any technological constraint, as Com Hem had asked for. This way of thinking was new and unfamiliar for both the agents and the development teams building and maintaining Com Hem’s other internal systems, which is why consultants, with expertise in user experience, were used. Insights from the field study were then discussed among a small team consisting of two interaction designers and two art directors. The goal was to brainstorm ideas to help solve the customer service agents biggest problems of the current software tools/systems. The results from the brainstorming sessions was a proof of concept (PoC) prototype (see figure 1) that was built to show four common usage scenarios to the Com Hem management team 2012-03-29. The PoC was successful and a pre study was started on the 8th of May 2012. The project got its budget and official start on the 12th of September 2012. On the 6th of November 2013, Alo went live and the old CRM system was decommissioned.

![Figure 1: The first prototype of Alo](image-url)
5 Related Work

There are several research papers and theses on the topic of integrating Agile with User-Centered Design/User Experience Design/Interaction design. Most of the articles and books on the subject highlights that "user-interface design and usability are largely overlooked by the agile processes" [20, p. 4], as an explanation to why the processes can be hard to reconcile. Most articles also describe similarities like working iteratively. What follows is four articles that combined, have become the groundwork for the framework suggested in the results, section 9. The framework suggests how to improve a combined development process of UCD and Agile at Com Hem in general, and especially in the Alo development team.

5.1 Constantine & Lockwood 2002

Constantine and Lockwood suggests close collaboration between designers and developers and a minimal design upfront (MDUF) that includes: Overall layout, navigation and keeping a consistent look-and-feel [20]. To avoid inconsistencies in the interface, Constantine and Lockwood suggest initially creating the overall navigation architecture. Index cards are used as a tool for brainstorming solutions and prioritizing them. This, according to Constantine and Lockwood, will create a common understanding within the development team. However, the idea that the navigation architecture, overall behavior and look can be set initially, and not changed fundamentally, before implementation, is by definition a large design upfront. This method does not solve the just in time design that Agile development promotes.

5.2 Stefan Blomkvist 2005

Blomkvist [5] states that one of the key differences between Agile methods and UCD is that "...the agile approach is concerned with strategies for effective teamwork, project management and organization culture. In contrast, UCD is less concerned with these issues... [5, p. 239]" Another difference Blomkvist describes is that Agile in practice often overlook the need for usability experts and interaction designers and also suggests that "The lack of awareness is probably due to an inadequate grasp of the importance of usability."

Blomkvist also points out similarities between UCD and Agile:

- A preference of face-to-face communication.
- Motivated team members.
- Clear responsibility.
- When people are happy and have a constant, reasonable workload they are most productive.
- The team should on a regular basis reflect on how to become even better at working.
- Even though all team members should be a part of most activities, both
UCD and Agile respects individual expertise.

- People build software, processes don’t.

**To bridge the gap between Agile and UCD, Blomkvist suggests that:**

- Users and developers should be co-located.

- The team should capture needs with user stories instead of use cases. User stories takes context of use and user goals into account, by asking questions like "would a real user do that?" and "does it fit into the users workflow?".

- Usage has to be put in context to understand the users goal rather than just the need for a certain functionality.

- Let users test the the system while in implementation in real working conditions i.e. "beta testing", not just when a feature is completed and delivered.

- Involve users more often. Also realize that the customers are often not the real users. It can be very misleading to think of customers as representative users.

- Design deliveries and presentations should only be done if it helps communication and understanding within the team.

- Automated tests and acceptance tests can not alone evaluate usability, qualitative usability tests with real users are also needed.

- Prototypes, not just the current version of the implementation, should be used and tested.

- The focus should be working and usable software.

- The users should decide on what features they need, with the assistance of usability experts.

- To improve communication, people should work in pairs, not just developer-developer but also developer-usability expert.

- Set usability goals, in numbers, and follow up results.

- Incremental releases to users could be skipped by usability testing prototypes that cover more than just one new function.

### 5.3 Desirée Sy 2007

I the paper: Adapting Usability Investigations for Agile User-Centered Design [3], Sy describes a scenario very close to the experiences of the early Alo development:

> During the design phase, we would rapidly iterate on key designs for a release, using formative in-house usability testing to direct the
re-design of prototypes (...). We would then describe the validated
designs as feature specifications, and pass them on to development
to be incorporated into the implemented code.

In theory, the analysis and design phases preceded the implemen-
tation (...). However, in practice, developers would begin coding at
the onset of a project without waiting for feature specifications. The
result was that the implementation of some features would begin be-
fore they were designed. To combat this tendency, we investigated,
designed, and validated well in advance, often conducting usability
investigations almost a full release ahead. However, this led to writ-
ing many unused or out-of-date feature specifications, since we could
not anticipate all planning issues (such as business goals). [3, p. 115]

To combat this, Sy’s team adopted just-in-time design:

Because developers are working on only a subset of features at
one time, and interaction designers are designing the same subset,
this also means that any features that require careful UCD work
receive it. Since everything in the product must be done, traces of
half-complete features don’t impede the user experience [3, p. 117].

Sy’s team adopted "Cycle Zero" and describes different ways of usability in-
vestigation depending on what type of feature or product that was supposed to
be built: Refining current feature/product, completely new feature/product,
the currently developed feature, investigating a new market. Sy describes these
week-long activities as mini waterfalls of UCD. During the investigative cycle
zero, developers focused on architectural programming or features that needs
little or no (interface) design. Sy suggests designing one cycle ahead and gather
requirements two cycles ahead (see Figure 2).

Figure 2: Cycle Zero, by Desirée Sy

This was not the entire solution: "We could complete small designs in this
timeframe, but complex designs required more than four weeks to finish." [3, p.
120]
Sy further suggests: "Interaction designers are trained to consider experiences holistically, so breaking designs into pieces — especially into pieces that do not initially support workflows — can be difficult at first, but it is a skill that comes with practice [3, p. 120]." Sy’s project is however developed in a Scrum like process with set cycles. Kanban on the other hand does not say on what date a feature will be delivered, therefore making it harder to perform what Sy calls design chunking. Design chunks are "...design components that can be prototyped iterated, and validated within Agile timeframes." [3, p. 122]

Sy suggests finding basic functionality in design chunks and start designing them first because they are fundamental to the use of the interface and do not change over time (e.g. navigation). Users may see several functions as one high-level action. If so, try to find a testable chunk that is the common known denominator of said functionality. In Sy’s case, different algorithms for movement of objects within the digital drawing application. The micro interaction with fundamental design chunks such as movement or a search function could be tested with colleagues instead of users if they are basic and not too context specific. Later stage design chunks would be tested on real users and realistic tasks. The Alo team has found that the problem with delayed usability testing with real users is that the product is nearly finished and fundamental redesign is very expensive at this stage. Sy suggests building realistic tasks by asking real users (via telephone) of typically one or more workflows, 2-3 weeks before the actual usability test. Sy also suggests using a board with index cards displaying findings from usability tests to get a clear sense of what the issues were and how they would be addressed.

5.4 Salah et al. 2014

In "A Systematic Literature Review for Agile Development Processes and User-Centered Design Integration" [31], Salah et al. reviewed 71 papers on the subject published between 2000 and 2012. Their findings were among other things: "...incremental Agile development is translated into sliced or "feature by feature" development for design, which can result in a UI that is disjoint, piecemeal and lacking a holistic, coherent, and overall structure and vision" [31, p. 5], suggesting a pre development stage of interface design before beginning to implement the solution.

Salah et al. suggests that problems with prioritizing user-centered design activities should be resolved by having separate backlogs for UX and development. This was the case at the beginning of the Alo project, but didn’t encourage more communication between usability experts and developers, rather it widened the gap, since they are not working to solve the same problems.

Salah et al. further suggests that the entire team (i.e. developers, testers, interaction designers, art directors, usability experts and so on) should strive for:

- A shared understanding of the users.
- A shared design vision.
- Attend daily meetings.
- Sharing knowledge with developers on how to perform usability tests.
Since usability testing can involve features from more than one sprint Salah et al. suggests:

- Performing shorter, lighter, usability tests and heuristic evaluations.
- Having access to users was facilitated by planning ahead and using recruitment firms.

Salah et al. also reports that one team solved time shortage in improving design after testing, by dedicated sprints/cycles for only improving the interface. The Alo team has tried this approach but it seldom got prioritized by the product owners, since it is not business critical (e.g bugs that violate business rules are more important).

6 Method

This section describes both the approach of reaching the objectives for this master thesis as well as the development process of Alo from the authors perspective. The reason behind this, is that the thesis was performed and written at the last six months of three years of developing Alo. The description is mainly from the UCD perspective, that was the authors expertise and main role in the development team.

The method consisted of, in chronological order:

1. A literature review.
2. Retrospective analysis of the Alo development from 2012 to 2015.
3. Creating an anonymous survey for the developers, to get their view on UCD in Agile development.
4. Documenting how the Alo team had performed usability testing from 2012 to 2015, and presenting the result to the IS/IT department.
5. Involving Developers in Usability Testing (planning, building, performing and evaluating).
6. Constructing a framework for further integration of UCD in Agile development at Com Hem.

6.1 Literature Review

The literature review had two goals:

1. Find successful ways of combining Agile development style with User-Centered Design thinking.
2. Review different types of prototyping and testing techniques and evaluate what would fit the Alo teams agile development style.

The result of the literature review can be found in section 5, Related Work.
6.2 Retrospective Analysis of the Alo Development Process

With the insights gathered from the literature review, a retrospective analysis of previous development was performed. The analysis served as the foundation for building a framework for UCD in Agile software development in general, and specifically at Com Hem.

6.2.1 Defining Requirements for Alo

The goal with Alo was to replace several legacy systems with one web based interface (with a Java based back-end). This meant that the functional requirements became a combination of both existing functionality in the legacy systems as well as new functionality that was missing.

From the onset of the Alo project, Com Hem had two web based tools to store requirements:

- Jira [32], a project tracking software/web service for user stories.
- Confluence [33], a wiki software/web service for more long term documentation.

Two business experts from the customer service department were assigned as Product Owners (PO) for the Alo team. They were the source of most requirements for Alo and helped write user stories mostly together with two usability experts: An Interaction Designer (the author of this article) and an Art Director, who both worked in the development team. Since none of the people defining the requirements had much experience with writing user stories, this was a learning experience, meaning that the initial requirements often described a solution in long specification documents (i.e. several A4 pages). Over time, the specifications were discarded in favor of lighter user stories that contained a template of headings, namely:

- **Background.** Optional description of why this story had any relevance or describing the current problem.
- **Purpose.** Described with the syntax: "As a [type of user], I want [goal to achieve] so that [reason]". This is a common way of writing user stories described in more detail in the section 7 chapter.
- **Definition of Done/limitations.** What should and should not be working when the story was completed.
- **How to test the story.** A step-by-step script on how to test if the story was correctly implemented.

Besides the description there were attachments, mainly consisting of wireframe or fully designed images of the interface.

The goal with the user stories was that anyone in the team, product owners or any stakeholder could understand why the story would be implemented and be able to test it when it was implemented.
The main problem with user stories was that they tended to have long descriptions containing technical solutions and more functionality than manageable in one sprint or even several sprints. This lead to the adoption of "epics" as a way of defining the bigger picture. Epics described the need of the costumer service department, like "creating a customer card" or "Creating a customer support ticket". Multiple stories (MVPs, MLPs and LPs) were gradually created and implemented to continuously cover the needs of the users and learn along the way.

6.2.2 Sketching on Paper

The proof of concept prototype that initiated the project had been an interactive wireframe built with a prototype tool (Axure RP). To save time, static images from the prototype were used as a basis of discussions within the team and with the product owners. A couple of months into the project the focus would be to start with analog sketches on paper (e.g. Figure 3) or whiteboards to make all involved in the requirements process feel more engaged and prone to make suggestions of improvements [34].

![Analog paper sketches of the search function](image)

**Figure 3:** Analog paper sketches of the search function

6.2.3 Creating Digital Wireframes

When the product owners had approved the design, static wireframes were added to the user stories to describe the interface solution. Wireframes for Alo were created with the prototyping tool Axure RP version 6.5 and 7.0, see Figure 4. Axure supports not only static wireframes but also creating highly functional interactive prototypes used in proof of concepts and usability testing.
6.2.4 Usability Testing

From the onset of the Alo development in 2012, prototypes and usability testing played a key role for both short term goals (i.e. the user stories in a sprint) as well as a long term vision of the Alo interface. The prototypes varied in fidelity from hand sketches on paper to coded prototypes. Usability tests were performed with all the different prototypes from said range, although most tests (19 in total) were performed on interactive wireframe prototypes with the basic style guides applied (i.e. colors/layout). The usability tests performed in the Alo project closely follows the method described in Handbook of Usability Testing, by Jeff Rubin [18].

For Alo, four categories of prototypes were used:

1. Paper prototypes. Mainly used as discussion material with different stakeholders and product owners.

2. Static wireframes or design images. Used for live comparison tests (e.g. "which of these designs do you prefer?") or for electronic surveys (e.g. "What does this icon mean to you?").

3. Interactive wireframes. The main tool for usability testing of either micro interaction or entire workflows, spanning multiple views (e.g. creating a new customer and ordering broadband and tv services).

4. Coded prototypes. Used when testing parts of the interface that needs large amounts of interaction and real data from underlying systems (e.g. search functionality with auto suggestion feature).

Most usability tests were performed on a computer setup matching the agents own meaning a PC keyboard and mouse and a 24" external monitor. To doc-
ument the tests, screen recording software was used (ScreenFlow for the most part, and Tobii studio at two separate occasions) with an external web camera to record test subjects reactions. The tests were planned and conducted by one person (mostly the author of this thesis) and annotated by one other person. The product owners and developers were present at about half of the test sessions. At one of the first tests, the scrum master of the team, suggested role-playing instead of describing a fictional scenario to the test subjects. This proved highly effective and triggered the customer service agents usual behavior (i.e. interacting directly with the test leader as if over a phone line), making the test significantly more realistic. The results of the tests were condensed into a presentation of findings accompanied by short excerpts from the recorded sessions. In three of the usability tests, three different developers were given the task of being the test leader. This was very enlightening according to the developers. This also helped in transferring knowledge about usability testing to the rest of the development team.

6.2.5 Using high-fidelity prototypes

Less focus on design for users meant that they demanded more of the prototype and found more inconsistencies like labelling/wording and less questions about design in general. Since Alo was a high functionality interface, a lot of things were conveyed with colors, contrast and layout. This meant that the prototypes in testing needed to have the basic color scheme. Since the prototyping tool, Axure RP \(^3\), supports design elements, the visual design was taken from the actual implementation, therefore not taking extra time when building prototypes.

6.2.6 Usability Testing with Expert Users

The author of this thesis created a framework on how to perform usability tests with expert users [35], and did a case study on said framework. The subject was explored because performing usability tests with expert users is hard [36] and can hinder innovation [37]. This is because expert users tend to prefer the tools they have mastered and new tools can, initially, reduce their performance, among other negative effects.

6.2.7 Using a Pilot Group

A pilot group consisting of 5-10 agents were formed and given access to the early version of Alo in December 2012. The goal with the pilot group was to get early user feedback from real usage. However, this proved difficult the first couple of months, because the pilot users could not complete entire workflows, therefore sticking to the old systems instead. The pilot group was used to test the Minimum Viable Product (MVP) features that were developed. Feedback from the pilot group was then taken into consideration in creating the Minimum Lovable Product (MLP) for said functionality.

\(^3\)http://www.axure.com/
6.2.8 Surveys With Customer Service Agents

Three yearly surveys, among the agents, was performed to find what application or system they like the most. This was a part of measuring one of the goals with Alo (see section section 4.3): "Being the most loved internal tool with the best user experience for the customer service agents."

The statements in the survey were (translated from Swedish):

1. The system is easy to use.
2. The system is fast enough to not effect my work negatively.
3. The system is logically built/designed.
4. The system is reliable.
5. The system has all necessary functionality.
6. The system helps me by presenting relevant offers to the customer.

All questions/statements had to be graded from on a scale from 0, meaning "I do not agree at all", to 5, meaning "I fully agree".

6.2.9 From Scrum to Kanban with XP

The Alo team followed the build-measure-learn cycle from Lean and built Minimum Viable Products [27] (MVP). However, in the Alo project, Minimum Lovable Product (MLP) and Lovable Product (LP) were used from the onset, trying to reflect the Alo team's desire to create a lovable interface. The concept of MVP was later added, to acknowledge that the team also delivered value both by creating something the team learned from, e.g. back-end development, and the MVP:s released to the pilot group, for user feedback.

The Alo development used a Scrum process (see Figure 5) but the two-four week sprints were abandoned because it did not fit into the "Lean startup" thinking that Com Hem strived for in the Alo development. Some of the development teams working with the legacy systems at IS/IT still followed scheduled monthly releases which meant that the Alo team had dependencies. Kanban was adopted to visualize the workflow and reduce the amount of work in progress (WIP), see Figure 6. Pair programming was adopted from the Extreme Programming (XP) methodology to improve code quality and increase knowledge sharing between developers [38].

6.2.10 Roles in the Alo Development Team

The roles in the development team, and their time spent developing Alo, from spring 2012 until spring 2015 are described in the table below. The number of developers are accumulated over those three years. Average number of developers were lower.

The product owners worked closely with the interaction designer and art director to create user stories. Most stories were written from a customer service
agent’s perspective. The interaction designer and art director then added wireframes and design sketches to illustrate user interface implementations. The user stories were then presented to the entire team to discuss goal, value and specific details concerning implementation. The stories went through a series of iterations before being put into the work backlog in prioritized order. The Kanban cards were printed and put on the physical Kanban board as a physical manifestation of the Kanban board in the digital tool Jira.

6.3 Survey With Developers

In the spring of 2015 an anonymous survey was conducted to find what part of the user-centered design that were most problematic and rewarding for the developers.

These are the questions in the survey sent to the nine developers that
had developed Alo:

1. To what degree have you been involved in defining the requirements for Alo?
2. Would you like to be more involved in defining the requirements?
3. To what degree have you been involved in creating user stories in Jira?
4. Would you like to be more involved in writing user stories?
5. To what degree have you been involved in the design of the interface?
6. Would you like to be more involved in the design of the interface?
7. What is the quality of the Alo user stories on average?
8. Is there anything missing in our user stories in Jira?
9. What are your feelings on wireframes of the Alo interface?
10. To what degree have you been involved in usability testing?
11. Would you like to be more involved in usability testing?
12. Wireframes are visions that stretch months or years into the future. Our agile development style focuses on what is implemented right now. Has this contrast in thinking effected the development negatively in any way?

The number of developers in the Alo development team had increased from nine to about twelve in the spring of 2015, however these new developers were not included in the survey.
6.4 Involving Developers in Usability Testing

As suggested by Spool [39], involving developers in usability testing could help improving the overall usability of Alo as a product and aid communication about usability related questions within the development team. Four of the nine developers took the role of test leader during usability testing and all nine developers had participated as note taker during the usability tests that were performed between 2013 and 2015.

6.5 Limitations

To more accurately capture and discuss the communications and interactions within and outside the Alo team, as a part of the design process, an external observer would have been preferable over the authors own anecdotal experience. The discussions regarding those matters in this thesis should therefore be seen as the authors personal reflection.

Since the development team had little experience with analytics tools (e.g. Google Analytics) and the fact that users behavior is partially driven by the customer, there has been little analysis of quantitative usage data regarding navigation within the Alo interface in real life situations. Knowing not only what features are used, but also how the features are used in connection with each other, micro interactions, and user flow through the interface, could be a great source of feedback in learning how to prioritize user stories in the future.

7 Agile Development Processes

Developing software and digital services can be complex; often involving a lot of people within and outside a company; and can therefore be time consuming. In todays fast evolving world of digital services, time is one of the most important factors. Being the first to release a product to the market can be the difference between success and failure. A structured way of development can therefore be seen as a competitive advantage against a company’s rivals [40].

Developing software and digital services has become a substantial part of what companies in both the private and the public sector do. This is reflected in the 2012 report from Statistics Sweden (Statistiska Centralbyrån) showing that the second most common profession among men in the age of 16-64 is some form of developer [41] and the eighth most common profession for both men and women [42].

7.1 Lightweight Development Processes

One of the most classic development processes is known as waterfall. The waterfall process was the most commonly used in the 1960s and 1970s. It consists of a series of sequential steps where each step in development results in a handoff to the next step [43, p. 86]. The major drawbacks with the waterfall process is that it is inflexible, slow and does not respond to changes. The system can not be
tested and validated until almost the entire project is completed [44]. According to the Standish Group [45] only 20% of delivered features in software development is used often, and 50% are seldom or never used. This means that a lot of what is produced can be seen as waste [19]. The more common development frameworks of today (2015), such as Agile, involve an iterative approach [40] i.e. an incremental development cycle to refine the software or system in order to accommodate for change and reduce waste.

Agile is an umbrella term for different lightweight software development methods that share a set of values and principles. Agile was popularized in 2001 with the release of the “The Manifesto for Agile Software Development” [46] and has four main values [19]:

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

The Agile manifesto also has twelve principles [46]:

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

Agile is, as mentioned above, an umbrella term guarded by principles. There are many specific Agile development methodologies, among the more commonly practiced is Scrum [47] with over 56% adoption, followed by a Scrum and Extreme Programming (XP) hybrid with 10% adoption, soon followed by Kanban with 5% adoption, according to The 9th Annual State of Agile Survey [47, p. 9].

This is a short description of the most prominent features of Scrum, Kanban and XP [38]:

**Scrum** A product owner is responsible for building a prioritized backlog with problems to solve. The development is performed in short intervals, called sprints, of two to four weeks. Each sprint begins with a sprint planning session where the development team picks the top item from the backlog and breaks it down to chunks that can be developed during one sprint. A demo is held at the end of the sprint, showing stakeholders what has been implemented. Finally, a retrospective meeting is held at the very end of the sprint where the development team and product owner discuss on what to improve in the next sprint. The insights for the retrospective meeting is then used at the beginning of next sprint.

**Kanban** Scrum limits the number of things that the team is working on by just adding enough to be completed within a sprint. Kanban is similar to Scrum, although it limits the number of things that are being worked on at the same time with a work in progress (WIP) limit. Kanban does not set a time limit for how long one thing can be worked on. One of The main advantages with Kanban over Scrum is that it is more flexible. What Scrum and Kanban both have in common is the board that visualizes the work in progress.

**XP** Extreme Programming, XP, is a process with very few rules and empathizes teamwork with its core concept; pair programming. Coding together increases the shared understanding among developers and increases the quality of the code as well as collective ownership and responsibility for the code. Test driven development and automatic testing are two of the most important tools for the XP developers.

7.2 User Stories

User stories are a way of defining required functionality of a system, described from a users point of view [48, 49]. It explains what the user wants to achieve and how this may benefit the user. User stories does not tell the developer how to implement a solution rather describing the expected outcome, which is the biggest difference compared to typical requirement documents.

According to Gothelf [26, p. 122] the user story can either: Deliver value to the user, deliver value to the team or just teach the team something new. This suggests that learning is equally important as delivering more functionality. This last addendum by Gothelf is rather important in the eyes of the author of this thesis, because there are often back-end implementations that neither stakeholders or users sees. In a strict sense, these learning stories can therefore
be seen as not important to implement (since they deliver nothing to the user), thereby making the development team miss opportunities of exploration and innovation.

8 User-Centered Design

"UCD represents the techniques, processes, methods, and procedures for designing usable products and systems, but just as important, it is the philosophy that places the user at the center of the process [18, Handbook of Usability Testing p. 12]."

In User-Centered Design, (UCD), the development of a product or service, the focus lies on the user and the entire experience during the product’s or service’s lifecycle. UCD shares the iterative design thinking of Agile development processes.

8.1 Making Sure it’s Usable by Testing it

There are several ways to ensure that a system and its user interface (UI) is usable. One of the most commonly known method is usability testing. However there are several more tools with their own strengths and weaknesses. These are some of the most common usability tools:

Heuristic Evaluation "Heuristic evaluation (...) is a ‘discount usability engineering’ method for evaluating user interfaces to find their usability problems. Basically, a set of evaluators inspects the interface with respect to a small set of fairly broad usability principles, which are referred to as the ‘heuristics.’ [50]"

Interviews Interviewing users is a good complement to understand the users needs. Interviews can be performed by them selves or as a part of a debriefing during usability testing [51].

Surveys "Surveys can be used at any time in the lifecycle but are most often used in the early stages to better understand the potential user. An important aspect of surveys is that their language must be crystal clear and understood in the same way by all readers, a task impossible to perform without multiple tested iterations and adequate preparation time. Again, asking people about what they do or have done is no substitute for observing them do it in a usability test [18]."

Qualitative Usability Tests Usability testing, sometimes called Qualitative usability tests (to differentiate from statistics based usability testing) is a controlled form of testing the usability of a product or service [18] that involves representative users that perform realistic tasks. Usage is observed and data can be gathered in different ways like note taking or video recording of the session.

Analyzing Real Usage Statistics With tools like Google Analytics and Google Tag Manager, real usage can be collected as statistical data or single usage. This is a good complement to qualitative usability tests.
8.2 Weaknesses With Qualitative Usability Tests

Qualitative usability tests has some commonly known shortcomings and biases [52] that must be taken into consideration when planning, performing and evaluating the test:

**Hawthorne effect** Users being observed tend to act and perform in a way that they normally would not do. It is also called the observer effect. This could be avoided by using real usage analytics as a complementary measurement.

**Task-selection bias** The users will think that they will be able to do something (in the interface) if the test leader asks it. It is therefore hard to test a feature without hinting that it exists.

**Social desirability** Users will tell the test leader what they think is what the test leader wants to hear. This makes answers from questions regarding likeability of an interface hard to evaluate.

**Availability** Usability tests will be performed on those who are available at the time of the test. They may not be the representative users.

**Honorariums** The quality of the data can be reduced if the motivation for performing the usability test is money.

"If you’ve asked me about it, it must be important" Users may be inclined to create an answer even though the user does not have an opinion.

**Note taking** Users can see when the observers are taking notes, thinking that they have done something wrong. This may cause them to try to undo it. A solution can be to take notes constantly or remotely.

**Recency & primacy effects** When letting the user evaluate different solutions or designs, the last shown or tested tend to get the highest rating because it is at the top of the users mind. Reordering the designs tested can help reducing this bias.

**Confirmation bias** [23] Searching for results that confirm the hypothesis. This is a typical bias that the person performing usability tests can be effected by. Having multiple people observe the usability tests and using metrics (e.g. counting the number of times users do a certain thing in the interface) can help reduce this bias.

**NIH syndrome** "Katz and Allen describe the phenomenon that stable groups (i.e. people that know each other very well) have a tendency to reject new knowledge from outsiders, for example colleagues from another department or consultants. Their suggestion is to involve users as well as managers early in the development, to show that their feedback is both valuable and crucial for building a usable interface [35]."

**Functional fixedness** "Functional Fixedness means that it is hard to solve problems in novel or different ways once you have learned one way to solve it (...), thereby increasing the learning curve [35]."
9 Results

These are the results of the master thesis. First, separate results are described, then a suggested framework is presented, building on the separate results.

9.1 Result from Survey with Developers

The main results of the survey showed that 6 out of 9 developers felt that they only were partially involved in defining the requirements and 5 out of 9 would like to be more involved. The results in its entirety can be seen in the Appendix section 13.

9.2 Results from Surveys with Customer Service Agents

The results from the survey, both in 2013 and the latest one performed in 2014, showed that Alo was the overall most liked by the customer service agents (Alo’s users). Figure 7 shows the result from 2014. The figure shows the score of five criteria with 0 meaning really bad and 5 meaning really good. "Sales help" functionality in Alo was not released to the common user, only small features released to the pilot group, which explains the low score.

Figure 7: Survey from 2014. Ratings for the top three systems according to the users, the customer service agents.

Focusing on the users had payed off not only in creating a tool that reduced handling time, but also being the most liked.

9.3 Usability Testing High-Functionality Interfaces like Alo

One problem with building high functionality interfaces (hi-fun) like Alo is that they demand that the user not only understands the interface, but also the processes of why something is done in a certain way. This means that a new solution may break the user’s mental model, even though the interface elements...
are understood. This means that users with extensive knowledge and experience with the systems being replaced, can have a harder time accepting said change.

A separate research paper was written on the subject of usability testing with expert users. These are the results from the article "Usability testing with expert users can undermine innovation - Developing a framework for testing high functionality interfaces" [35].

- Involving expert users and managers in discussing a redesign reduces aversion to change.
- Describing changes made in workflow and interface before performing the usability test helps creating a constructive dialogue about problems with the current systems and how to improve the design.
- Coded prototypes work well with expert users since it helps them explore and thereby both learn the new interface and unlearn old habits.

9.4 Problems in the Alo Development

This list represents the greatest usability and user-centered design challenges that has come up during the three years of development from the start of the Alo project in 2012.

- Conveying the value of usability tests to organizational developers from the customer service organisation. Organizational developers felt out of the loop when going directly to the customer service agents instead of people higher up in the organisation. Delegating more of the responsibilities to the developers, as is a natural progression when adopting agile methods [38], further shifted influence and control. This was not how development had been done in the past, causing initial friction between the development team and the customer service organisation. The root to some of the frustration could be attributed to the implementation of the COTS customer relations management system (CRM) in the spring of 2009 [53].

William Hudson [54] opines that:
"Another issue is that representative users (or user representatives) are frequently chosen for the wrong reasons. They may be;

- domain experts;
- either popular or unpopular with their colleagues (depending on the method used to select them);
- either popular or unpopular with management (depending...);"

Jakob Nielsen [55] puts it: "One of usability’s most hard-earned lessons is that ‘you are not the user.’ If you work on a development project, you’re atypical by definition. Design to optimize the user experience for outsiders, not insiders." Desirée Sy [3, p. 123] on the other hand argues that not all functionality needs to be tested on the real users by saying "We reserve external users to test only mid- to late-stage design chunks, and the focus of those usability tests is on validating design goals that can
only be determined by an actual user."

- Not being included in the education program for all agents (the agents introduction to Alo) made it hard to understand how new users perceived Alo and what parts of the system that were good and what needed more attention from the development team. This may also have been because the implementation of the COTS system demanded a customer service specific education, not involving the IS/IT department in Stockholm.

- The geographical distance between the IT-department in Stockholm and the customer service in Härnösand, causing communication problems and making it hard to perform usability tests as frequent as the team wished.

- Cuts in travel budget for our product owners (not being able to travel to Stockholm for months). Björkholm and Brattberg [38, p. 84] suggests having a proxy for the product owners that is always co-located with the development team. They further suggest that the product owner(s) should preferably choose this proxy him- or herself.

- The team does not record usage of all current functionality. Fully using analytics tools like Google Analytics/Google Tag Manager, could resolve that problem.

- Legacy systems are forming the requirements (meaning: being stuck in old way of building patchwork solutions).

- Can not deliver functionality in full scale, to all users, until an entire workflow is built (e.g. can not start moving a customer in Alo and then finishing in legacy system X).

- Promising a specific set of functionality (or entire workflow) to be delivered on a certain date when using Kanban. This was mainly a problem when the number of features could not be reduced to keep the deadline, because of business rules. A specific example was being able to plan an education sessions long in advance for all 800 users, not knowing if all features for a workflow would be built in time.

- In one planning meeting a developer noted that "The problem now is that since the PoC is pretty much complete and you have uncovered most of the problems, the development now is more or less a matter of following a specification. You have taken out the fun part." By letting the PoC process stretch over a couple of weeks and with just one developer and one user experience expert, we had in fact created a specification. Since programmers are problem solvers by nature, this meant that we had taken all the joy out of implementing the feature. Or put another way: We had reduced most of the intrinsic motivation factors.

### 9.5 Successes in the Alo Development

These are the key success factors in the development of Alo according to the team summarized by the program manager of the team.

**Close collaboration** The most important success factor of Alo was the close collaboration between IS/IT and the Customer Service Organization. This
made it clear that IS/IT wanted to build a tool that made work easier and faster for the agents, not like the introduction of the previous COTS system in 2009.

**Full ownership of system** Since Alo is an in-house developed tool, not a bought proprietary system, Com Hem has the freedom to design solutions as they see fit. This also means that the Customer Service Organization owns the system, controlling both how their agents work and how the tools work.

**Value driven development** Building what is needed the most and releasing it when it is ready, instead of functionalities released on a deadline.

**Continuous delivery** Being able to deploy continuously, when needed instead of on a fixed schedule means not only that bugs can be fixed the same day they are found, but is also gives the ability to develop incrementally.

**Build-measure-learn** Using the build-measure-learn loop that Lean principles advocates.

Below is a list of some of the concrete results in the Alo development:

- Meeting the users more often than any other IS/IT team.
- Reduced average handling time per call.
- The ability to toggle off features that are deployed to production, meaning that the team could deliver and test things, bit by bit.
- Shortened the education in system handling, for new customer service agents, from five days to two.
- In a survey among the customer service agents (Alo’s users), Alo became the number one most liked system, among five, both in 2013 and 2014. The most recent survey was performed in 2014. See Figure 7.
- The lead time for developing functionality was shortened and developed at lower cost than all previous introductions of new IT systems at Com Hem.

### 9.6 Remote Usability Testing

The hardware and software that the agents use, have not permitted remote usability testing so far, thus limiting the number of live call observation and usability testing because of time and budget constraints. Remote usability testing would enable the team to perform usability testing on a weekly basis. Scheduling agents for usability tests would remain a problem, However, this could be solved by instead having the usability experts, in the development team, on standby.
10 Future Work

Remote usability testing and live viewing of the usability tests, would be two of the most interesting techniques to explore if future work.

11 Discussion

One of the main reasons that Agile does not consider user-interface design could stem from the fact that: "...no representatives from the human-factors, interaction design, or usability communities were invited to participate in the formation of the Agile Alliance", as Constantine [20, p. 4] points out.

Another obstacle was that the backlog had grown over time with hundreds of old user stories that were either duplicates or irrelevant. The team had to go through and delete more than 100 user stories to make the backlog manageable. This is most likely a result of not using techniques like user story mapping and not having a synchronized process for design and development. The lack of a pre-populated template in user stories in Jira may have been the cause of the varying quality of user stories, even though the team had agreed on the one described in section 6.2.1. This made the backlog even more opaque.

11.1 Bringing Agile and UCD Together

User-Centered Design (UCD) is about exactly what the name says: Designing with the user in focus. UCD does not specify how an interface should be implemented. Agile methods on the other hand is focused mainly on satisfying the customer, as the first principle of the agile manifesto clearly states. This is the key difference between UCD and Agile, because the customer is almost never a representative user of what is implemented. The customer knows the business needs, and may view usability issues as unnecessary. This means that the only way to really integrate UCD in Agile development is when the customer puts the users’ needs first and their own second. This could explain why projects that described in the literature can end up with such different results when integrating UCD and Agile.

The long term thinking and rigorous research in UCD stands in contrast to the learning by doing attitude in Agile development, and is not easily reconciled. Most books and papers suggest giving the initial usability research a head start and either progressively integrating the two processes completely, or keeping the (interface) design one cycle ahead. Few of the sources explain in detail how this should be accomplished. The reason may be that they acknowledge that every project or company is different from the next, that there is no general solution that applies to all.

The following list is a suggested framework for integrating UCD in Agile development mainly for the Alo team. However, they are written in a more general form for anyone to use, especially when building high functionality software or systems used as a professional tool. The list is a combination of lessons learned (successes and failures) from three years building Alo, combined with insights from the literature study:
Colocation of the entire team One of the most important success factors of Agile development is having the entire development team and stakeholders in the same office [56]. Colocation means that the decisions on design choices can be made just in time, not requiring documentation. In Alo, this became painfully evident when budget restrictions kept the product owners in Härnösand, not being able to travel to the development team in Stockholm. This resulted in a reduced delivery speed because of the increased test backlog and reduced not only the frequency of communication, but also its quality.

Self organizing team The development team needs to be self-organizing [46], to deal with the problems themselves and encourage responsibility. Let the team negotiate with external parties themselves. This is in line with the idea that managers in agile projects should coach the team instead of controlling it.

You are not the user Neither you, stakeholders or the client are the real users [55] of what you are building. You may use the thing you are building, but if you are involved in creating the it, you are not a representative user.

Observe what the users do Listen to what the users has to say, but focus on what they do. This could be achieved both by sitting next to the user and observing behavior via some form of analytics/statistics.

Aim for a synchronized process In the beginning of a new project, use what is called “cycle zero”. It means to build the design one sprint ahead of development. After a couple of sprints, evolve towards a synchronized process [26] and create design just in time for the development. Developers are problem solvers and like specifications as little as anyone else. As one of the Alo developers said during a pre sprint discussion: “You have already solved the problem, so we can basically just implement right now. You have taken all the fun out of it.”

Dedicated UX designers for each team Avoid creating a separate UX team that handles many different teams and projects at once [57]. There should be at least one usability expert within every team.

Set expectations Set expectations by performing a “way of working” workshop [45] where you describe your expectations on the other team members and learn what the other team members expect of you.

Create vision prototypes Build vision prototypes [58, 59], but keep them separate from the Agile development processes and keep them off Scrum or Kanban boards. These prototypes are used to create a shared vision within the team and as a communication tool with outside stakeholders.

User Story Mapping USM is a technique for creating a big picture of what is being built [60] and in what order, but not getting into details until it is time to design and implement the details.

Shared vocabulary/terminology A great way to reach a shared understanding within a team, as well as with outside stakeholders, is to define the shared terminology [61], e.g. what parts of the interface is called. This is preferably created within a workshop and documented within a wiki that all involved have access to.
User stories Refine user stories together with the help of all competences of the development team represented, not just by the customer or product owner who may have created them. Agreeing on a template and content of a good user story (headings, definitions, images/videos, how to measure success etc.). This stands in contrast to what Mike Cohn [49] says: Stories should be created by the customer. However, the Alo team discovered that several competencies was needed to create a good user story.

Involving developers in usability testing Involving developers in usability testing makes them care about usability more and get a better understanding of the users. Make assumptions, together with the team, before usability testing [39].

Watch the usability tests in real time Let the development team and all stakeholders watch the usability tests [62] in real time, preferably in the same room. This is a good way to get a shared understanding of the most important usability issues.

Live style guides Create a living style library for the actual design and micro interaction (e.g. how a date field should look, validate, etc.). This makes it easy to create coded prototypes [26].

Use coded prototypes When performing usability tests where parts of the interface that has complex interactions (e.g. search functionality), use coded prototypes. This helps the users (typically expert users) explore the interface and learn on their own [35].

Involving users early Talk with users and all stakeholders, before, during and after development. Listen to their problems, not their solutions. This is an efficient way to resolve the “not invented here problem” (NIH) [63].

Learning has value on its own Every story should deliver value, however that value can be that the development team has learned something [26]. It does not mean that the end user has to get some new functionality.

Test often, with a purpose Usability tests can be small and focus on an input field as well as an entire workflow. Performing usability tests to resolve internal debates about small design choices as well as for large workflows [37].

Pair program and pair design Pair program and pair design as much as possible [5]. This is a good form of knowledge sharing. Pair programmers with designers, even if it seems inefficient at first. This helps create shared understanding of both coding and designing issues.

Sketch often Sketch on paper and whiteboards as much as possible when discussing problems. Let anyone sketch, not just designers [26]. Make everyone visually show what they are thinking. Do not let them get away with drawing half a rectangle and speaking for 15 minutes.

12 Conclusion

User-Centered Design can indeed be combined with Agile development methods, however it requires the entire development team as well as stakeholders to
believe in the benefits of both methodologies and accept concessions from their respective, more strict, definitions.

The framework presented in the result section can be used by any development team trying to combine UCD and Agile, even though it is specifically designed to improve the development of Alo at Com Hem.

13 Acknowledgments

I want to thank Com Hem, and especially my program manager Mikael Rask, for providing references and input on what areas of the Alo development process to explore.
References


[8] Dealbook: Blockbuster files for bankruptcy (September 2010)


[34] of Health & Human Services, U.D.: Prototyping (November 2015)


[52] Sauro, J.: 9 biases in usability testing (August 2012)


on Software Engineering Advances. (2013) 359–363


1A. To what degree have you been involved in defining requirements for Alo?

Not involved at all
Sometimes involved
Involved most of the time
Always involved

1B. Would you like to be more involved in defining the requirements?

I would prefer to be less involved
I like my current level of involvement
I would like to be more involved
2A. To what degree have you been involved in creating user stories in Jira?

- Not involved at all
- Sometimes involved
- Involved most of the time
- Always involved

2B. Would you like to be more involved in writing user stories?

- I would prefer to be less involved
- I like my current level of involvement
- I would like to be more involved
3A. To what degree have you been involved in the design of the interface?

- Not involved at all
- Sometimes involved
- Involved most of the time
- Always involved

3B. Would you like to be more involved in design of the interface?

- I would prefer to be less involved
- I like my current level of involvement
- I would like to be more involved
4. What is the quality of the Alo user stories on average? 0 = really bad, 10 = really good

5. Is there anything missing in our user stories in Jira?

- I have not seen a pattern in what is missing. It varies from story to story.
- Acceptance test description (gotten much better lately).
- Business value.
- The purpose of the story.
- Joy.
- No (3 participants).
6. What are your feelings on wireframes of the Alo interface?

- Blue: They kill my creativity
- Yellow: They help me a little in my work
- Green: They hinder my creativity a little
- Orange: They help me a lot in my work

7. To what degree have you been involved in usability testing?

- Blue: Not involved at all
- Green: Sometimes involved
- Orange: Involved most of the time
- Brown: Always involved
9. Wireframes are visions that stretch months or years into the future. Our agile development style focuses on what is implemented right now. Has this contrast in thinking effected the development negatively in any way?

- Not really, although I have had the feeling that the wireframes have been dated and not based on the current information.
- Sometimes, when there are quick-wins done in code (temporary, for now solutions). They can stay forever if we are unlucky.
- Not really.
- Sometimes a bit confusing when developing the interface.
- I sometimes feel that I lack the vision when I'm implementing a smaller part/component.
- Seldom.
- No. (3 participants)