From Mob Programming to Mob Development: User-Centred Design in Collaborative Software Development

Victor Anderfelt
Uppsala University
Uppsala, Sweden
v.anderfelt@gmail.com

ABSTRACT

Mob programming is a collaborative software development method that has gained increasing attention in both industry and research. While the focus of mob programming is on the benefits of teams programming together, there are also potential benefits for other aspects of the software development process. However, there is a lack of research on the use of the method outside the domain of programming. This study explores user-centred design (UCD) in mob programming through a case study of three software development teams at Sveriges Television, a Swedish public broadcasting company. Results show that the teams use the method for a variety of tasks in their daily work, calling for a rebranding of the method to *mob development* to encompass the broader scope. The integration of UCD is analysed through the principles of user-centred agile software development. The results indicate that a revision of these principles is needed to include the cross-functional and social factors that mob development adds to the software development process.

Author Keywords

Mob Programming; Mob Development; UCD; UCASD; Collaborative Software Development; HCI practice.

INTRODUCTION

In the wake of an increased interest from software development businesses to acknowledge the importance of usability and user experience (UX) design, a need for discussions about collaboration and multi-disciplinary software development teams has been expressed from the industry [1, 2, 29]. The field of Human-Computer Interaction (HCI) practice further acknowledges the growing integration of user-centred design and research practices in industry. Ogunyemi, Lamas, Lárusdóttir and Loizides [25], for example, have found in their systematic mapping study that HCI practice research has not only focused on design tools, methods, and contexts, but also on the importance of collaboration and team communication. These two standpoints; the industry's desire for multi-disciplinary collaboration, and the research field's stance on collaborative HCI practices, demonstrate a need for cross-functional collaborative methods for HCI practitioners and developers.

One such collaboration practice is mob programming [37]. Mob programming is a software development approach born from agile development philosophies such as extreme programming [4]. Like pair programming [35], mob programming involves simultaneous work at one computer from several people. However, unlike pair programming, mob programming involves more than two people, often whole teams, working together at the same computer.

The interest in mob programming has increased during the past few years, as seen by the discussion surrounding how to best conduct mob programming both in Sweden and internationally [5, 11, 12, 24, 33]. As it is described by Zuill [37], one of the leading proponents of mob programming, the method focuses on the coding aspects of software development projects. However, he also argues that the approach can be applied to all stages of a software development project, from discovery to implementation and beyond. As this study will focus more on this broader use of the method, I have chosen to refer to the method as *mob development*.

Most of the current research on mob development focuses on programming [e.g. 3, 6, 18]. Although some researchers have explored the integration of UX professionals and software development teams using agile frameworks [8, 13], there is a need for understanding how collaborative methods like mob development incorporate HCI-work. Previous research on the integration of UX professionals in agile teams has concentrated on user-centred design (UCD) as part of HCI knowledge and practices [14, 32], a perspective taken in this paper as well. UCD is used as a broad term encompassing a focus on end-users as well as practical aspects such as user evaluation and refinement of design concepts [7]. This makes UCD an appropriate perspective for analysing HCI practices in agile processes, especially in the field of User-Centred Agile Software Development (UCASD). This study uses the framework of principles of UCASD presented by Brhel, Meth, Maedche and Werder [7], who produced a foundation for the integration of UCD and agile software development based on a thorough literature analysis. By exploring UCD in mob development through a case study of three teams from the interactive department at Sveriges Television (SVT), a public service broadcasting company working actively with mob development, I aim to answer the following research question and two sub questions:

How can user-centred design be integrated in mob development?

1. *How is mob development used at SVT?*
2. *What principles of UCASD are supported by mob development?*

BACKGROUND/RELATED LITERATURE

This chapter presents mob development and the research and practitioner literature that covers the method. This is followed by an overview of the literature on UCD and UX in agile software development. Lastly, the theoretical framework used in this paper is presented.

Mob Development

Mob development is a relatively new method, however, the original term, mob programming, was first coined by Hohman and Slocum [17]. The authors were attempting to simplify and streamline their coding practices following a transition to extreme programming at their company. The method, and specifically the term mob programming, did not gain traction until Zuill [37] presented his team's experience of working in the method-specific collaborative fashion. According to Zuill, the method was born from extreme programming and an agile development approach. He describes the method as an approach to software development where an entire team works together on the same thing. The team sits at the same computer with a large projector or screen, where one person is the "driver", controlling the keyboard and mouse, while the others are "navigators", discussing and explaining what the driver should do. Since the method evolved from the experiences of Zuill and his colleagues, it is primarily developed for programming and code testing. However, he suggests that mob development has potential for being used for all aspects of a software development project, from early discovery phases to deployment.

Apart from Zuill's experience report there have been some other instances of mob development being tried in practice. Wilson [36] describes how he and his team went from using pair programming to mob development. According to Wilson the method was useful for solving complex tasks, but not as successful for more complicated problems where the solutions were already known. Similarly, Buchan and Pearl [9] applied mob development in a software development team and discovered both benefits and challenges with the method. Some benefits were increased productivity, consistency of the code design and broader knowledge of the system in the entire team, while some challenges were that the learning curve was steep and getting used to the method was difficult. The team also found social friction occurred more often when using the method. These experiences were also present in Boekhout's [6] introduction of mob development in two less experienced teams. Unlike the other studies, however, Boekhout reports that mob development was successfully used in non-software development settings.

Apart from experience reports, research conducted on mob development is sparse. Balijepally, Chaudhry and Nerur [3] conducted a literature study covering benefits and risks associated with the method, such as less technical dept but greater risk for social friction, as well as suggestions for future research. The authors argue that although early adapters of the method see value in the method there is a need for the research community to validate those claims. Considering the relative short time mob development has been discussed and used, Balijepally's et al. call for more research is sound. However, their study, as all literature on mob development, focuses on the programming aspect of software development process'. There is an apparent lack of research concerning non-programming aspects of mob development and the method could therefore benefit from being investigated from an HCI perspective.

User-centred Design and User Experience Design in Agile Software Development

Understanding how UCD can be incorporated in mob development can be difficult considering the lack of literature on the method. However, several studies have been conducted concerning UCD and UX design integrated with other agile processes and practices that can help illustrate the standpoint of relevant research.

One way of integrating UCD in agile frameworks is through collaborative methods. Collaborative methods in software development, other than mob development, like for example pair programming, are commonly used and have been thoroughly studied [15, 31, 35]. Some researchers have focused on collaborative methods other than those focussing on programming. Ramanujam and Lee [30] developed a collaborative agile framework and investigated its effects in a large multinational company. They propose a framework for collaborative work on an organizational level focused on team-level collaboration which resulted in several strategical benefits. Not only did the collaborative framework simplify the requirements process in the case company, but it also improved other processes like user story mapping and development. Apart from direct collaboration between and in teams, managerial and organizational support has been shown to be important for UCD to be integrated successfully in agile teams [16].

Bruun, Larusdottir, Nielsen, Nielsen and Persson [8] studied UX professionals' roles and responsibilities in agile teams. The UX professional role is decidedly broad, covering responsibilities related to several disciplines and development phases [8]. Bruun et al. found, after interviewing 10 employees at an IT company, that the role has responsibilities that are more cross-disciplinary compared to what previous research has identified. The UX role includes sales and business development responsibilities in addition to the more traditionally UX related responsibilities of e.g. interaction design, prototyping and user research. The authors argue that the case company they studied integrated UX disciplines well through an active involvement of the UX professionals in sales meetings and business development. However, these newly identified responsibilities emphasize customer needs over user needs. The authors discuss the choice between user- and customer needs and conclude that there are situations where working user-centred is not possible.

Although the professional UX role has a broad connection to various parts of an organization, several studies have shown that integrating UX processes and methods might not always be easy. Kuusinen, Mikkonen and Pakarinen [21] investigated the interaction of user experience design and software development activities. They identified several challenges and issues with the integration of the approaches that are in line with the findings of Bruun et al. [8]. The identified challenges related to communication and frictions between management and development processes. Ferreira, Sharp and Robinson [14] also investigate the integration of UX and agile development. In line with Kuusinen et al. [21], Ferreira et al. [14] find communication between divisions and roles to be important factors, however present a more positive outlook on the situation by suggesting improvements for the integration. Organizational structure and support for communication and coordination between teams and roles are shown to be important for design practices to be embedded in agile software development processes successfully [13, 14, 20].

A different approach to the integration of UX practices in agile development is presented by Øvad, Bornoe, Larsen and Stage [26]. Through action research the authors let software developers learn, apply, iterate, and improve upon usability tests. The study shows that in some cases the sole responsibility for certain UX tasks need not necessarily be on the UX professional role, a conclusion reached by Ferreira, Noble and Biddle [13] as well.

The field of UCASD takes a less practice-based perspective to the definition and responsibilities of UX professionals. There are, however, two exceptions to this. Patton [27] presents an experience report where he describes the loss of usefulness and usability of products when following a strict extreme programming approach. By introducing interaction design to the day-to-day software development process Patton argues that the resulting products are of higher quality. Although Patton does present some recommendations for incorporating UCD in the agile processes these are mainly of practical nature. Similarly, Lárusdóttir, Cajander and Gulliksen [22] study what user evaluation practices are used in agile software development teams, concluding that user feedback is often collected informally before the actual development begins.

Unlike Patton [27] and Lárusdóttir et al. [22] most other literature on UCASD takes a broader standpoint, taking an explorative look at UCASD. In a separate paper Cajander, Lárusdóttir and Gulliksen [10] address the challenges when adopting user perspectives specifically in Scrum projects. They highlight responsibility, documentation, and user centred activities as well as organizational and contextual setting as factors that affect how well user perspectives are adopted and incorporated in Scrum projects. In line with these findings Wale-Kolade, Nielsen and Päivärinta [34] also found that context and situational factors are important but require more research. Communication and knowledge sharing between roles is important for the integration of UCD and agile software development [10, 19], however Cajander et al. [10] do not discuss more collaborative approaches to the integration. Da Silva, Martin, Maurer and Silveira [32] conducted an extensive systematic literature review of the UCASD field to pin-point common practices, suggestions, and challenges in the literature. The authors present several similarities between the studies, taking a sprint-based standpoint in presenting suggestions for how to integrate UCD and agile software development. From a research perspective the authors establish that more empirical or experimental studies are needed.

Based on the literature overview presented here the studies on UX and UCD in agile software development show that there are several factors to successfully integrating user-centred perspectives and practices in agile approaches. The role of the UX professional, communication, organizational support and context-dependent factors are some elements that have been shown important, however much of the literature focus on the practices and tools used in software development and UCD. Additionally, even though the literature concentrates on practices or results in practical suggestions, there is an obvious lack of consideration for collaborative methods as a tool for easing the integration.

Principles of User-Centred Agile Software Development

Bhrel et al. [7] recognized the need for an even broader perspective on UCASD, finding previously conducted literature reviews to be lacking in quality. The authors point out three major shortcomings: (1) the reviews do not encompass all dimensions of UCASD, (2) the reviews do not offer generalizable results and (3) the reviews do not provide sufficient quality assessment of the reviewed literature. With these shortcomings in mind, the authors present a thorough analysis and summary of UCASD literature by conducting yet another systematic literature review, aiming to answer the research question: “which principles constitute a user-centred agile software development approach?” [p. 2, 7].

Bhrel et al. [7] included 83 papers in their review, after a four-stage quality assessment process. The papers were divided into dimensions, coded iteratively and then analysed, ultimately resulting in five principles (Table 1). Based on their literature review, the authors found a lack of unanimous findings in the two dimensions they call “technology” and “people/social”, leading to the five principles only covering the dimensions the authors call

Principle	Description
1. Separate Product Discovery and Product Creation	User-centred agile software development should be based on separated product discovery and product creation phases, with an emphasis on research- and design upfront.
2. Iterative and Incremental Design and Development	User-centred agile approaches should support software design and development in short iterations and in an incremental manner.
3. Parallel Interwoven Creation Tracks	Design and development should proceed in parallel interwoven tracks. The principle stresses finishing design tasks before development but acknowledges the importance of cross-functional teams to improve the cooperation between UCD experts and developers.
4. Continuous Stakeholder Involvement	Stakeholders should be actively involved in user-centred agile approaches early on and should remain involved throughout the entire development process to collect input and feedback.
5. Artifact-Mediated Communication	Tangible and up-to-date artifacts should be used to document and communicate product and design concepts and should be accessible to all involved stakeholders.

Table 1. The five principles of UCASD by Brhel et al. [7]

"process" (including principles 1, 2 and 3) and "practices" (including principles 4 and 5). The principles cover the, according to literature, important aspects of UCASD making them an appropriate support for understanding UCASD. In this paper the principles will be used to discuss the integration of UCD in mob development, however the lack of principles focusing on social and technological aspects of UCASD is acknowledged.

METHOD

This study will follow the case study approach [23], collecting data through semi-structured interviews. The choice of case study is based on the assumption of ontological realism, that there is knowledge to be gained by exploring and researching phenomena in their real contexts. This is accomplished by applying the principles presented by Brhel et al. [7] to "the reality" of mob development. The case studied in this paper has a history of working with mob development for several years and has worked to integrate more than just programming in the method, which allows exploration of mob development in its real-life context. As Lazar, Feng and Hochheiser [23] explain,

> … case studies use careful analysis of carefully selected subjects to generate interesting and novel insights, ideally with an eye on developing general principles that might facilitate understanding of other cases. [p. 153, 23]

The lack of previous research on mob development and the integration of non-programming tasks further cement the choice of case study. By carefully analysing the case, and developing general results based on the findings, the resulting insights may help create an understanding of how UCD can be integrated in mob development.

The Case of SVT

The case company studied in this paper is SVT, the Swedish national public service television company. SVT's services span from television broadcasting to online news and streaming services. In this study I worked with three teams from the news department at SVT's digital branch SVT interaktiv (SVTi). The three teams work with developing and maintaining the news- and sports website and apps. The members of the teams are used to working together both in person and remotely and can choose work processes and methods freely. This freedom has led to the teams' dedication and engagement in finding the best process for their teams. The teams all use mob development as a method, to varying extents, every day. The content in the mob sessions differ between the teams, but all teams have had some experience adapting mob development to both programming tasks and design related tasks. This makes SVT and especially the news department an appropriate case subject for studying user-centred design in mob development.

Data Collection

I conducted nine semi-structured, qualitative interviews with members of the three different teams. The roles of the interviewees were one or a mixture of UX designer, art director (AD), product owner (PO), developer (DEV), and tester (TEST). Five of the interviews were conducted in person at the company office, while four were conducted remotely through Slack. The interviews were held in Swedish, lasted 40 to 60 minutes and were recorded with consent. During the interviews I followed an interview guide and followed up interesting subjects not included in the interview guide as they appeared. The interview guide consisted of 17 questions and follow-up questions: two questions were related to the interviewee's role and their team's general workflow and software development process, nine questions were on mob development, how they use the method, what strengths and weaknesses the method has, and six questions were on UCD, how they apply UCD practices and methods as well as if and how they apply it in mob sessions. In many cases the interviewees answered the questions with examples from their current or past projects, which helped give context to their expositions. These examples were carefully anonymized to ensure confidentiality.

Thematic Analysis

The data analysis process followed recommendations for content analysis [23, 28], and was conducted digitally. Content analysis, as described by Lazar et al. [23] and Patton [28], is similar to what can be referred to as thematic analysis in that it aims to reduce large data-sets to smaller, yet rich and descriptive, patterns and themes. Content analysis is well suited for analysing rich interview data to generate general principles, in line with the case study approach [23].

The process followed several steps beginning with a read-through and notation of the transcripts. The transcripts were transferred to the digital sketch and interface design tool Figma, for easier visual access of the text data, and were re-read and coded. The coding was deductive, based partly on the theoretical framework, like "design upfront" based on principle one [7], and partly on concepts related to mob development, like "Empathy/Perspectives". When new codes emerged later in the process, the transcripts were re-read to see if they fit in earlier sections.

After the iterative coding process, the codes were compared and considered based on possible converging categories, as suggested by Patton [28]. The grouped codes were compared to the principles of UCASD [7] and categories were created from the principles. These categories were complemented with additional categories that were not covered by the principles such as "mob development, not mob programming" and "how is mob development integrated?", resulting in ten categories in total. However, after analysing the categories' external heterogeneity there were several aspects of them that were too similar [28]. This led to another iteration of the categories resulting in nine final top-level themes. The data gathered from the interviews was recorded and transcribed in Swedish. The coding and thematic analysis was therefore also done in Swedish. Clarifications and translations to English were done for the quotes included in the report.

RESULTS

This chapter is divided into sections based on three main findings. The first section, *Defining Mob Development*, outlines what, according to the informants, mob development as a method is and how it is implemented. The second section, *Mob Development and Agile Processes*, presents the informants' description of how the method fits in with other processes, and how processes are affected by the method. Collaboration was shown to be an important aspect of the method. Even though it is touched upon in the first two sections, I deem it important enough to be highlighted in its own section due to its frequent occurrence in the interviews. The last section, *Collaboration and Sociality*, outlines these social and collaborative aspects of mob development.

Defining Mob Development

A recurring theme during the interviews was the use of mob development for more than just programming. This became apparent in different contexts during the interviews but was in many cases also directly referred to. The informants described what constitutes as mob programming, including stakeholder involvement, the use of artefacts and other methods, and the effect organizational support has on the use of mob development.

Mob Development, not Mob Programming

When describing mob development, the interviewees talked about several different aspects of the method, such as group collaboration, knowledge sharing, and programming, among others. All the informants had a general idea of Woody Zuill's [37] explanation of the method, something that came up despite not being a part of the interview guide. However, most of the interviewees described this definition of the method as too narrow for their needs. One of the interviewees explained how they define mob development: "*It doesn't have to be sitting and writing code or sitting in front of a screen. A lot of our mob development is us sitting on the couches in our mob station and discussing the problem.*" (PO-1). As the informant describes, the teams often refer to group-based, non-programming tasks such as design studios and workshops as mob development. The broad definition of the method is also described by the following informant: "*But in our team we've extended that concept to not only be about coding, and it's not really test-driven either, we try to apply the same way of thinking to all aspects of system development.*" (DEV-1). According to this informant, *"[E]verything from coding to, what do I know, maybe planning a user test, in some case we have sketched on design sketches together ...*", counts as mob development. What the informant describes, similarly to the others, is that mob development is used by the teams for many different tasks, and in many stages of the software development process.

For these reasons, many of the interviewees used the term *mob development* rather than mob programming. It is safe to assume that they did so to strengthen the idea of the mob including more than just programming and being a method for more than just developers. This was expressed by one informant: "*I think that it's important to say mob development and not mob programming, and that's because I'm a designer, because my background is not developer, rather the opposite.*" (AD-1). The term, mob development, highlights the importance of solving different tasks, not just programming. However, exactly what tasks should be solved in a mob vary according to the interviewees, ranging from user story mapping to mock-up creation.

Some of the informants suggested that mob development is less effective when performing administrative tasks like documentation, pointing to the participants' individual documentation styles. Others argued that the method was appropriate for any type of work. The different teams used the method to varying degrees which became apparent when discussing what types of problems can be solved using mob development. One informant explained that mob development works best when: "*you need to solve a more complex problem where you need several approaches to minimize any errors.*" (AD-1), while another informant said it works best when: "*There is an immediate problem, we don't know what's causing the problem and we don't know how to solve it, when we need all competences we think we need to solve this together.*" (PO-2). The two quotes show disparate opinions of when mob development is best used.

There are apparent differences in what types of problems and tasks the informants suggest could be solved with mob development. However, all informants agreed that mob development involves a group of people solving one problem. As one informant put it: "*[mob development] is actually just a type of collaboration. And it feels like a design studio could be a mob as well. Because you do it together.*" (PO-1).

Stakeholder and User Involvement

According to the informants, another benefit of using mob development is the ability to include people from other teams, users, or other stakeholders directly. The physically and temporally immediate nature of mob development means collaboration and insights without worrying about lead times. One informant expressed it like this: "*When [a stakeholder] becomes a part of our team it's much easier to just, like `what is he thinking of?´, because he is sitting next to us.*" (DEV-2). Including stakeholders in the mob, according to this informant, lets the stakeholder become a part of the team. The importance of including stakeholders in the mob can be related to principle four, *Continuous Stakeholder Involvement* [7]. Principle four expresses the importance of stakeholders being involved in all aspects of the software development process. Mob development allows for stakeholders to be involved, however as one informant puts it: "*the mob can be a good tool for building good relationships with stakeholders, but I don't think it is the main purpose with [mob development].*" (DEV-2). In other words, mob development can help improve communication and build relations in the organization but might not always be the main reason for using the method.

Artefacts and Methods

Some of the informants stated that having the right tools for the problem at hand is important for mob development. Whiteboards were mentioned by most of the interviewees as important, and especially when the problem is related to UCD. One participant said, "*[W]e actually work with these boards. And, anything from just sketching something or having a short workshop where you put up some post-its, or physically moving post-its to prioritize things.*" (AD-1). The informant describes how important the physicality and tangibility of whiteboards is to their team's work process.

Although some of the informants described using tools, and most described using whiteboards, some of the interviewees questioned the value of the tools in mob development. One participant reflected on the use of user-story maps: "*But since [updating the user-story maps] is done every day, actually twice per day, during check-in and check-out… The board doesn't become as important, rather it's the discussion that is important.*" (PO-1). Another informant presented opposing thoughts in their reflection on the use of whiteboards in general: "*since you make decisions together, you make choices together, I think that the tools we have around us might help us remember what choices we've taken. ... So, it becomes some sort of collective memory.*" (DEV-1). In other words, there are differing opinions about the value of tools in mob development.

Communication in mob development is often assisted using tools such as whiteboards and user-story maps. According to the fifth principle, *Artifact-Mediated Communication*, this is central for UCASD [7]. However, while the principle emphasizes communication and documentation, mob development adds a need for active participation through the artifacts. Due to the collaborative nature of mob development, the artifacts become an integral part of the method, not just mediating communication between roles and teams. This leads to a problem that was touched upon during the interviews: the participants in the mob need to know how to use the tool. One participant expressed the difficulties like this: "*There is a threshold there that is challenging which is, partly that everyone doesn't know the tool that you are working in, Sketch or Figma or whatever you want to use.*" (UX-2). When doing UCD-related work through mob development, not knowing the tools may become an issue, interrupting the workflow.

Supporting Mob Development

Many of the interviewees expressed the freedom their teams had regarding how to best meet the strategic goals. As an informant expressed it: "*we start with the strategic goals ourselves and find which one we can work best with.*" (PO-1). In other words, the teams have the freedom to choose what to focus on. This is also the case regarding *how* the teams work. Thus, the teams choose to use mob development for both UCD and programming. However, it is not enough for a team to have the freedom to use mob development for UCD if the organization does not support the team, and the team does not have the right attitude. This was discussed by one of the interviewees: "*Well, I would say it's a willingness from the team ... [UCD is] difficult to integrate in [mob development] if neither the organization nor the team wants it.*" (UX-1). A willingness to adopt mob development for more than just programming is required from both the organization and the team members.

Defining Mob Development Summarized

The interviews highlight the interviewees' definition of mob development as more than just programming. The method is used by all the three teams but to different extents, and for different tasks. Artefacts and other tools are used in combination with mob development, but apart from whiteboards there is no consensus whether specific tools are necessary for all teams. The tools are used for communication but are also a more integrated part of the work practice. Lastly, mob development requires support from the organization to be practiced successfully.

Mob Development and Agile Processes

The informants describe both strengths and weaknesses in using mob development for UCD during different stages of software development processes. This section presents the opportunities and challenges, expressed by the informants, that arise when using mob development during different project phases. Furthermore, the informants discuss how UCD is conducted, and how mob development helps facilitate design work.

Discovery, Delivery and Design

Many of the informants discussed *discovery* and *delivery* as two phases during their projects. Discovery was described as an exploratory phase, sometimes referring to it as a research phase. The research can be both technical and user-centred, including user studies, interviews and observations, or more technical investigations of the feature being implemented. Discovery also included planning and structuring goals, and formulating solutions to problems. Delivery was explained as more of an implementation phase, where most of the development and design is conducted. The use of mob development for UCD during the different phases appears to vary a lot depending on what stage of the project the team is currently in.

Some informants found mob development more challenging to utilize during the discovery phase. One informant discussed situations where mob development is easier to use, where there is a clear idea of the problem, and continued by saying, "*The opposite would be explorative phases where the problem isn't apparent yet, where there is more insecurity and questions.*" (PO-2). Conducting research in mob development was tackled differently among the teams. One informant described how research was conducted by the UX designer "outside of the mob": "*And there has often been requirements from outside of the mob like `hi, I have been out doing user tests and I found this out, think about that when you develop´.*" (DEV-2). The informant describes a situation where the UX designer has conducted research and presents the findings to the developers as a deliverable, rather than being part of the mob. Another informant suggested that: "*ideally [UCD] should be one sprint ahead, so that you have explored, understood, answered and straightened out most of the questions.*" (PO-2). A third informant stated that research was conducted collectively by the team in what they defined as part of the mob: "*some teams have that kind of mob where the mob is for the developers, and the designers and PO jump in and out and add their perspective. But we do it like, (two of the developers) are down there during the user tests*" (PO-1). These quotes show differences in how the teams choose to tackle mob development during discovery. UCD research can be conducted separately from the mob, in some cases one sprint ahead of development, or as part of the mob, including all roles in the process.

Likewise, the informants identified challenges with integrating design-related tasks in mob development, such as not knowing the tools being used or not having design skills. However, all of them expressed a desire to solve these challenges and successfully integrate UCD in mob development. As described by one informant: "*I mean, nobody wants a delivery structure where you receive UX and design in your lap and they say `this is what you'll do´, nobody likes that.*" (DEV-3). One of the solutions to this was doing design upfront, which could be built upon in the mob. One of the UX designers explained that having, "*[S]ome type of input value*" (UX-2), can make the processes easier. The same informant continued, saying, "*[I]t doesn't have to be exactly like it's supposed to be, but at least a guide to where we want to be which we can change*". The informant suggested that creating a first design to start with could help bridge the knowledge gap between the different roles.

Although the informants referred to the two phases as separate it was not always clear when one ended and another began. In some cases, the informants described a clear distinction between the phases: "*[The project] that we are working on right now is an example of where we have an ongoing phase of UX/UI exploration.*" (PO-2). In other cases, the distinction and separation of the phases was less clear: "*Well, I always think when it comes to the discovery, delivery phases, for me, whether it's mob or project based… It has often been pretty fluid either way.*" (DEV-2). Similarly, some informants described instances when design and implementation tasks were mixed in the mob: "*we translated the views, restructured a bit, and then we sat in the mob during the implementation, view-by-view.*" (UX-2). When integrating design and implementation in the mob, the interviewees still expressed a need for some design upfront, as seen in the previous quote.

Both research- and design upfront (collectively referred to as Design upfront by Bhrel et al. [7]) seem to be important for the teams. When performing tasks in mob development, Design upfront helps performing UCD related tasks in the mob. Design upfront is, according to the first principle, *Separate Product Discovery and Product Creation*, an important aspect of UCASD [7]. In some cases, the teams expressed difficulties integrating design tasks in mob development. This affirms the third principle, *Parallel Interwoven Creation Tasks*, which states that design and implementation should be separated in parallel, albeit interwoven, tracks [7]. However, in certain cases development and design are successfully integrated in mob development. This differs from the parallel aspects of principle three and presents an opportunity for design and implementation to be joint rather than parallel.

The integration of UCD in mob development varies among the teams depending on their needs and is something that the teams are investigating and experimenting with. The following informant explains, "*On one side UX and design* *are deliverables for development and on the other side everyone is doing everything together all the time, I think we have tried to find the middle.*" (DEV-3). This suggests that while the teams are trying to find a balance between doing no UCD versus doing everything in mob, they are not sure where the perfect balance is.

Agile User-Centred Design and Iterative Processes

The teams used agile processes, in various ways, utilizing Kanban, Scrum or by following iterative but less strict processes. The agile processes coloured the design processes, iterating on designs based on research, as one informant put it, "*[W]e discussed with [the stakeholders] a lot ... and checked with them what felt like almost daily.*" (AD-1). The use of mob development was also affected by the iterative nature of the team's work practices: "*one could have done that, [the UX designer] did user tests and then came back and talked about them. ... but we take turns doing them. So, that's why there are more iterations because we move the mob between them.*" (PO-1). By "moving the mob", the informant's team can be flexible in how to use the method, in this case moving the mob to include user tests.

Several other informants also presented examples of flexibility in their processes. One interviewee explained, "*I'd say that we probably flow back and forth in a few different processes depending on what we do.*" (DEV-1). The need to be flexible is based on the varied problems the teams encounter. The teams choose how they use different methods depending on which team members are available and what problem they are solving. Iterative research, design and development are natural parts of mob development, because of the method's roots in agile software development. Thus, the second principle of UCASD, *Iterative Design and Creation tasks* [7], is supported by mob development.

Mob Development and Agile Processes Summarized

The teams face different phases during their projects, referring specifically to discovery and delivery. The interviewees express challenges integrating research and design tasks in mob development during these phases but solve them through design- and research upfront. In some cases, design can be directly integrated with development.

Collaboration and Sociality

Mob development is a collaborative method, designed to increase the quality of the product being developed. It emphasizes social interaction and teamwork. This was shown to be important for the practitioners at SVT as well. The interviewees described cross-functional collaboration in mob development, expressed the importance of competence and knowledge sharing and presented several examples of the social aspects of the method.

Cross-Functional Collaboration

The three teams were all cross-functional, meaning they all included a mixture of developers, PO's, UX designers, AD's, testers and data analysts. As previously explained, the teams use mob development for a plethora of tasks. One reason for this being possible is collaboration between people with different roles. In many cases the informants even included this when defining mob development. As one informant puts it: "*I would say that for me [mob development] is when there is a cross-functional team, or at least cross-functional people, it doesn't have to be a team.*" (UX-2). In other words, mob development is defined, not only as collaborative, but as cross-functional.

Several informants also gave examples of different roles being part of tasks that traditionally are not included in the role's responsibilities: "*the rest of the team has definitely participated during these workshops, and been a part of the user tests and interviews, so we've done it as a team but with [the UX designer and AD] as leads for the work.*" (DEV-1). Several informants described examples of designers practicing collaborative design. One example of this is explained by this interviewee: "*we had a design-, not a sprint but more like a design studio ... the entire team and we developed ideas on paper, post-it white board level.*" (UX-2). Cross-functional teams are shown to be important for the integration of UCD through mob development. The importance of cross-functional teams is emphasized by principle three, *Parallel Interwoven Creation Tasks* [7].

The Importance of Competence and Knowledge Sharing

Even though mob development makes collaboration with different roles easier, getting the entire team to participate in UCD tasks is easier for some teams than others. One of the main challenges identified by the informants was a lack of knowledge regarding UCD and the tools being used. This was expressed by a UX designer, referring to digital sketching tools: "*if you're working together with different disciplines. That is, if there are developers, then maybe the tools may be a threshold*" (UX-2). Another informant reflected on the output of mob development if not all participants know what is being done: "*people have said that if it's like four front-enders and a back-ender in the mob then you'll be writing CSS. And if there's one designer and five developers it won't be design, it'll be programming.*" (DEV-2). This suggests that including a UCD specialist in a group mainly consisting of developers may not necessarily impact the result.

However, the informants also acknowledged the importance of including different roles in the same mob. This was partly because mob development was described as a good tool for knowledge sharing, and partly because not including different roles made it easier for the teams to forget the absent roles' perspectives. One informant, for example, said, "*It's noticeable that if you try to separate the mob, people will experience knowledge gaps.*" (UX-1). Another informant discussed knowledge transfer and different roles in mob development this way: "*You love to get that as a UX:er: `wait a second, this feels like a weird interaction pattern´ or `nobody has asked for this'. It's really nice to get that from [other roles].*" (UX-2). The informant refers to instances where they have received UCD related questions and remarks from other roles.

The Social Mob

One of the most important aspects of mob development, according to the informants, was the social aspect of the method. The informants discussed how the method helped improve communication in and between teams, reducing lead times significantly. One of the informants expressed it like this: "*It like develops quicker. Everyone is on board, everyone who can give answers is sitting there.*" (UX-1). However, the informants also explained that the social nature of mob development could lead to some frustrations and friction. One interviewee said, "*But as I said not everyone likes to work in mob because it's pretty socially exhausting as well. You easily become a group that does everything together. And it's important that you are comfortable with that.*" (PO-1). The informants also explained that for mob development to work as intended there needs to be a safe group culture. As one informant stated, "*There should be such a safe climate that you feel comfortable asking questions*" (TEST-1). Additionally, the informants emphasized a main strength of the social mob development: it helped the team members better understand the other roles' perspectives. One participant described it like this: "*as soon as we don't do it in mob the user perspective becomes overlooked ... just like you need to have an emphasis on your users you need to have an emphasis on your friends.*" (UX-1). Similar to missing out on UCD knowledge, not conducting UCD tasks in a mob can lead to missing out on UCD perspectives.

Collaboration and Sociality Summarized

Cross-functional collaboration is an important aspect of mob development. By having cross-functional teams, mob development helps integrate UCD in agile software development processes. Mob development is also an arena for knowledge sharing, leading to further improved integration. Lastly mob development is a social method which relies on good teamwork and safe group culture.

DISCUSSION

Mob development is presented in previous research as a collaborative method for software development, focusing on programming and technical problem solving. In this study I have investigated the use of mob development as a collaborative and cross-functional way to integrate UCD in agile software development processes. I aim to answer the research question: *how is user-centred design integrated in mob development?* This question is supported by two sub questions: *how is mob development used at SVT?*, and *what principles of UCASD are supported by mob development?*

Mob Development at SVT

Mob development is an important method for the three teams at SVT. The method is used, as first intended by Woody Zuill [37], for programming, but the method's uses cover much more than that. Similarly to what other studies have found, the method helps the teams solve complex problems and spread knowledge [36, 9]. However, the interviews show that the teams apply the method to problems over different competence areas and share knowledge cross-functionally. Furthermore, the interviews have shown that the method should not necessarily be limited to sitting as a group in front of a large monitor. The teams at SVT apply the same mentality to workshops, design studios and even user studies and research. This is possible because of the organizational support the teams receive and the freedom they have to choose how to reach their goals. The importance of organizational support is supported by several previous studies [10, 13, 14, 16, 20, 21, 34], which shows how organizations' trust in the teams affects the integration of UCD in agile approaches.

The need for a broader definition of mob development is apparent, and explicitly stated by the interviewees. As mentioned in this paper's introduction I have chosen to use the term *mob development* rather than *mob programming*. This was mentioned during the interviews as an important way to express the various ways the method can be applied, and for non-programmers to feel a sense of responsibility towards using the method. There is, however, a common acceptance among the interviewees that mob development is not perfect. Administrative tasks or documentation are examples of tasks that may not benefit from being performed in mob. But by letting teams have responsibility over their own work practices, mob development can be complemented by other, more individual or pair-based tasks as well. Despite the potential for mob development to facilitate UCASD, the use of the method outside the realm of programming has scarcely been acknowledged [6]. On the other hand, this study supports the value of integrating UCD practices in agile approaches, as suggested by previous research [22, 27]. To answer the first sub question: the teams at SVT show that mob development is an effective way of tackling complex problems outside the scope of mob *programming*.

UCASD Principles in Mob Development

Brhel et al. [7] presented five principles of UCASD based on findings from UCASD- and related literature. The findings in this paper suggest that mob development can both facilitate and be supported by the principles. However, some of the principles do not apply directly to the case of mob development, and some cases are missed altogether. This suggests that the principles need revising.

Mob development supports principle one [7] by encouraging design- and research upfront. Separating product discovery and product creation helps to keep the "solving one problem at a time" mindset in the mob and improves the involvement of developers in design-related work. Furthermore, design- and research upfront, and designing one sprint ahead of development supports principle three [7]. Despite this, the collaborative nature of the method and the cases where the interviewees describe developing and designing together in mob, suggests that principle three might not be entirely appropriate for mob development. Rather than focusing on parallel creation tasks, design in mob development may require some rough first sketches, with the need for further collaboration. Just as in earlier studies, the findings here suggest that an important part of integrating UCD in mob development is the cross-functionality of the teams [8, 13, 26, 30]. This further incentivizes the mix of development and design in mob development through joint activities across roles.

Principle two [7] is supported by mob development thanks to the iterative and agile nature of the method. As stakeholder involvement is an important part of mob development, principle four [7] is also supported. These principles would apply when using mob development without modification. The last principle, principle five, relates to artifact-mediated communication [7]. In mob programming, the physical setting and the tools being used are important. To some extent this is true for mob development as well. However, the broader definition of mob development refutes the explicit need for a monitor or computer. As the informants explained, design studios and workshops are also part of mob development. Interestingly, few tools were shown to be very important, apart from whiteboards. There is however a need for the participants to have knowledge of how to use the tools when they are necessary, for example when designing wireframes digitally. This suggests that rather than requiring specific artifacts to mediate communication, mob development requires all practitioners in the mob to collectively decide on what tools they need, and to learn to use them well, in order to be able to participate in the mob.

One aspect that is important for mob development, but which is not covered by the five principles presented by Brhel et al. [7], is the social aspect. Mob development is a collaborative method, with the aim of improving the quality of products by gathering input and perspectives from different people through active participation. Because of the method's social nature, mob development brings social challenges, which supports the findings of previous research [6, 9]. The findings in this study suggest that to mitigate these challenges the method relies on the groups having a safe culture and the participants being able to understand other roles' perspectives. Interestingly, Brhel et al. [7] present "people/social" as one of the four dimensions of UCASD. This dimension did not lead to any principles, however, due to the lack of support in their literature review. The results from this study suggest that this dimension should be revised and re-evaluated.

To answer the second sub question: mob development supports principles one, two and four without modification. Principle three is partly supported by mob development, but the method suggests the integration of design and development to be more prominent. Principle five can be supported by mob development, but should focus more on participation, rather than communication, through tools. A sixth principle focusing on the social aspects of the method is needed to fully encompass UCASD in mob development.

Reflections

This study is based on nine interviews with mob development practitioners. Despite the rich and descriptive data these interviews resulted in, the study could have benefited from a second source of data gathering. Originally, the study was going to include observations as a second data gathering source. Including observations could have enhanced the study by providing the benefits of methodological triangulation, which is recommended by both Lazar et al. [23] and Patton [28]. Because of the outbreak of Covid-19 during 2020, the observations were not possible to conduct. However, the data gathered through the interviews is of such quality that the results successfully portray the use of the method for UCASD, while highlighting the need for further research on the method.

CONCLUSIONS AND FUTURE WORK

In this paper I presented a case study of how user-centred design is integrated in mob development in three teams at SVT. The results from nine semi-structured interviews show that, unlike most previous studies show, mob development is a method that can be used for more than just programming. The method supports several principles of UCASD, specifically principles one, two and four, while presenting opportunities for revisions of principles three and five, and the addition of a sixth principle, to incorporate the social and more collaborative aspects of the method.

There is a need for more HCI studies to be conducted on mob development, and its uses outside the realm of programming. The collaborative and cross-functional nature of the method lends itself to new and interesting ways to incorporate UCD and agile software development, which would bring value to both the research field and mob practitioners alike. One aspect that was not covered in this study, but which would be interesting to further investigate, is how the role of UX practitioner is affected by more cross-functional and collaborative methods. Furthermore, I see a need to revise Brhel's et al. [7] principles of UCASD for the use of mob development. The social aspects of the method, connected to the people/social dimension of UCASD, especially need further investigation. Mob development is one new method birthed out of traditional agile approaches. Understanding this method can help understand how collaborative approaches improve the quality of HCI practice.

ACKNOWLEDGMENTS

I would like to thank SVT, and especially the news department at SVTi, for letting me conduct this case study. I also want to thank the many people whose brains I picked during my time at SVT. I would like to thank Helga Sadowski for the help she offered despite the outbreak of COVID19 (thank you Zoom). Lastly, I would like to thank my fellow student Michael for introducing me to mob programming in the first place.

REFERENCES

[1] Oluwatobi Akindunjoye. 2018. Reduce designer-developer friction: embrace UX collaboration. *Medium*. Retrieved January 22, 2020 from https://uxdesign.cc/reduce-designer-developer-friction-embrace-ux-collaboration-5627eaf55278

[2] Per Axbom and James Royal-Lawson. 2019. #223 Cross disciplinary collaboration with Becki Hyde. *UX Podcast*. Retrieved January 22, 2020 from https://uxpodcast.com/223-becki-hyde-cross-disciplinary-collaboration/

[3] VenuGopal Balijepally, Sumera Chaudhry, and Sridhar Nerur. 2017. Mob Programming – A Promising Innovation in the Agile Toolkit. In *Twenty-third Americas Conference on Information Systems*, 1-9.

[4] Kent Beck and Cynthia Andres. 2004. *Extreme Programming Explained: Embrace Change*. Addison Wesley Professional, Boston.

[5] Lea Kovac Beckman. 2018. 100% av teamet i en mobb i 12 månader — att ta mobbprogrammering ett par steg längre. *Medium*. Retrieved May 12, 2020 from https://medium.com/@leakovacbeckman/100-av-teamet-i-en-mobb-i-12-m%C3%A5nader-att-ta-mobbprogrammering-ett-par-steg-l%C3%A4ngre-29be5074a8f4

[6] Karel Boekhout. 2016. Mob Programming: Find Fun Faster. In *Agile Processes, in Software Engineering, and Extreme Programming* (Lecture Notes in Business Information Processing), 185–192. https://doi.org/10.1007/978-3-319-33515-5_15

[7] Manuel Brhel, Hendrik Meth, Alexander Maedche, and Karl Werder. 2015. Exploring Principles of User-Centered Agile Software Development: A Literature Review. *Information and Software Technology*. 61 (May 2015), 163–181. https://doi.org/10.1016/j.infsof.2015.01.004

[8] Anders Bruun, Marta Kristin Lárusdóttir, Lene Nielsen, Peter Axel Nielsen, and John Stouby Persson. 2018. The role of UX professionals in agile development: a case study from industry. In *Proceedings of the 10th Nordic Conference on Human-Computer Interaction* (NordiCHI ’18), 352–363. https://doi.org/10.1145/3240167.3240213

[9] Jim Buchan and Mark Pearl. 2018. Leveraging the Mob Mentality: An Experience Report on Mob Programming. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018* (EASE’18), 199–204. https://doi.org/10.1145/3210459.3210482

[10] Åsa Cajander, Marta Larusdottir, and Jan Gulliksen. 2013. Existing but Not Explicit - The User Perspective in Scrum Projects in Practice. In *Human-Computer Interaction – INTERACT 2013* (Lecture Notes in Computer Science), 762–779. https://doi.org/10.1007/978-3-642-40477-1_52

[11] Martin Christensen and Daniel Sundman. 2016. Fördelen med mob-programming: “Alltid någon som har svaret.” *Expressen Utveckling*. Retrieved May 12, 2020 from http://utveckling.expressen.se/blogg/mob-programming/

[12] Sam Fare. 2018. I did mob programming every day for 5 months. Here’s what I learnt. *Medium*. Retrieved May 12, 2020 from https://medium.com/comparethemarket/i-did-mob-programming-every-day-for-5-months-heres-what-i-learnt-b586fb8b67c

[13] Jennifer Ferreira, James Noble, and Robert Biddle. 2007. Agile Development Iterations and UI Design. In *Agile 2007 (AGILE 2007)*, 50–58. https://doi.org/10.1109/AGILE.2007.8

[14] Jennifer Ferreira, Helen Sharp, and Hugh Robinson. 2011. User experience design and agile development: managing cooperation through articulation work. *Software: Practice and Experience*. 41, 9 (July 2011), 963–974. https://doi.org/10.1002/spe.1012

[15] Shahla Ghobadi, John Campbell, and Stewart Clegg. 2017. Pair programming teams and high-quality knowledge sharing: A comparative study of coopetitive reward structures. *Information Systems Frontiers* 19, 2 (October 2015), 397–409. https://doi.org/10.1007/s10796-015-9603-0

[16] Jan Gulliksen, Inger Boivie, Jenny Persson, Anders Hektor, and Lena Herulf. 2004. Making a difference: a survey of the usability profession in Sweden. In *Proceedings of the third Nordic conference on Human-computer interaction* (NordiCHI ’04), 207–215. https://doi.org/10.1145/1028014.1028046

[17] Moses M Hohman and Andrew C Slocum. 2002. Mob Programming and the Transition to XP. 1–5.

[18] Herez Moise Kattan, Flavio Soares, Alfredo Goldman, Eduardo Deboni, and Eduardo Guerra. 2018. Swarm or pair? strengths and weaknesses of pair programming and mob programming. In *Proceedings of the 19th International Conference on Agile Software Development: Companion* (XP ’18), 1–4. https://doi.org/10.1145/3234152.3234169

[19] Johanna Kollmann, Helen Sharp, and Ann Blandford. 2009. The Importance of Identity and Vision to User Experience Designers on Agile Projects. In *2009 Agile Conference*, 11–18. https://doi.org/10.1109/AGILE.2009.58

[20] Kati Kuusinen. 2015. Task Allocation Between UX Specialists and Developers in Agile Software Development Projects. In *Human-Computer Interaction – INTERACT 2015* (Lecture Notes in

Computer Science), 27–44. https://doi.org/10.1007/978-3-319-22698-9_3

[21] Kati Kuusinen, Tommi Mikkonen, and Santtu Pakarinen. 2012. Agile User Experience Development in a Large Software Organization: Good Expertise but Limited Impact. In *Human-Centered Software Engineering* (Lecture Notes in Computer Science), 94–111. https://doi.org/10.1007/978-3-642-34347-6_6

[22] Marta Lárusdóttir, Åsa Cajander, and Jan Gulliksen. 2014. Informal feedback rather than performance measurements – user-centred evaluation in Scrum projects. *Behaviour & Information Technology*. 33, 11 (December 2013), 1118–1135. https://doi.org/10.1080/0144929X.2013.857430

[23] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. 2017. *Research Methods in Human Computer Interaction*. Morgan Kaufmann, Cambridge, Massachusetts.

[24] Karin Obermüller and Jeff Campbell. Mob Programming - the Good, the Bad and the Great. Retrieved May 12, 2020 from https://underthehood.meltwater.com/blog/2016/06/01/mob-programming/

[25] Abiodun Afolayan Ogunyemi, David Lamas, Marta Kristin Lárusdóttir, and Fernando Loizides. 2019. A Systematic Mapping Study of HCI Practice Research. *International Journal of Human–Computer Interaction* 35, 16 (November 2018), 1461–1486. https://doi.org/10.1080/10447318.2018.1541544

[26] Tina Øvad, Nis Bornoe, Lars Bo Larsen, and Jan Stage. 2015. Teaching Software Developers to Perform UX Tasks. In *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction* (OzCHI '15), 397–406. https://doi.org/10.1145/2838739.2838764

[27] Jeff Patton. 2002. Hitting the target: adding interaction design to agile software development. In *OOPSLA 2002 Practitioners Reports* (OOPSLA '02), 1–7. https://doi.org/10.1145/604251.604255

[28] Michael Quinn Patton. 2015. *Qualitative research & evaluation methods: integrating theory and practice*. SAGE Publications, Inc., Thousand Oaks, California.

[29] Loriah Pope. 2019. Cross-Functional Teams: Getting Engineering, Product, and Design on the Same Page. *Medium*. Retrieved January 30, 2020 from https://medium.com/swlh/cross-functional-teams-getting-engineering-product-and-design-on-the-same-page-555f981ce164

[30] Rohit Ramanujam and Ickjai Lee. 2011. Collaborative and competitive strategies for agile scrum development. In *The 7th International Conference on Networked Computing and Advanced Information Management*, 123–127.

[31] Fernando J. Rodríguez, Kimberly Michelle Price, and Kristy Elizabeth Boyer. 2017. Exploring the Pair Programming Process: Characteristics of Effective Collaboration. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '17), 507–512. https://doi.org/10.1145/3017680.3017748

[32] Tiago da Silva, Angela Martin, Frank Maurer, and Milene Silveira. 2011. User-Centered Design and Agile Methods: A Systematic Review. In *2011 Agile Conference*, 77–86. https://doi.org/10.1109/AGILE.2011.24

[33] Mikael Sundberg. 2017. A year of mob programming. *Medium*. Retrieved May 12, 2020 from https://engineering.klarna.com/a-year-of-mob-programming-e8cc7543ac2

[34] Adeola Wale-Kolade, Peter Axel Nielsen, and Tero Päivärinta. 2013. Usability Work in Agile Systems Development Practice: A Systematic Review. In *Building Sustainable Information Systems*, 569–582. https://doi.org/10.1007/978-1-4614-7540-8_44

[35] Laurie Williams and Robert Kessler. 2002. *Pair Programming Illuminated*. Addison-Wesley Longman Publishing Co., Inc., USA.

[36] Alexander Wilson. 2015. Mob Programming - What Works, What Doesn't. In *Agile Processes in Software Engineering and Extreme Programming* (Lecture Notes in Business Information Processing), 319–325. https://doi.org/10.1007/978-3-319-18612-2_33

[37] Woody Zuill. 2015. Mob Programming - A Whole Team Approach by Woody Zuill | Agile Alliance. Retrieved January 22, 2020 from https://www.agilealliance.org/resources/experience-reports/mob-programming-agile2014/

SUPPLEMENTARY WORK
Appendix 1 - Final themes and codes from the content analysis

Themes	Codes
Discovery, Delivery and Design	Discovery/Delivery Discovery hard Research upfront Design upfront Parallel design/implementation Design in development Research
Agile User Centered Design and Iterative Processes	Agile Iterative design Flexible processes Discovery in each sprint Data-driven design
Cross-Functional Collaboration	User perspective / UCD Cross-functional Cross-functional teams Cross-functional collaboration Collaborative design / UX
Stakeholder and User Involvement	Stakeholder involvement Cross-team collaboration UCD - several user groups
Artefacts and Methods	Tools – mob Tools - user story mapping Tools - white boards Tools – physical Remote Tools – digital Discussion over tool
Mob Development not Mob Programming	Mob - programming/development Collaboration UX/team responsibility Solving one problem Focus When? – Documentation When? – Always When? – Crisis
Supporting Mob Development	Organizational structure Organizational focus Organizational support Self-driven
The Importance of Competence and Knowledge Sharing	Technical challenges Competence Knowledge transfer
The Social Mob	Empathy/perspectives Communication Lead Times Mob – social

Appendix 2 - Interview guide

Welcome – thanks for being here and wanting to participate in this interview

In this interview you will be asked to talk about your roll, about how you and your team work generally as well as in mob programming sessions, and how you keep a user-centred focus.

Consent form – including if I can record/take notes

Introductory questions
Can you tell me about your roll, what you get to do as ___ and what your responsibilities are.

Could you describe your team's general workflow or a general system development process you follow?

Mob

Could you tell me what you think mob programming is? / Could you define mob programming?

- Who is part of a mob session?
- Could you tell me a bit about the tools you use in mob sessions?

In the process you described earlier – where does mob programming fit in? Why? Why not?

What is mob programming good for? When should it be used? When should it not be used?

If you think back to something your team has been working on recently – How did you use mob programming? When was it appropriate to use that method?

What is the most important thing for a successful mob session?

What is the most difficult thing about mob programming?

How do you think mob programming affects communication and teamwork between roles in your team?

UCD

Could you tell me about what user-centred design means to you? How would you describe/define user-centred design?

Do you think that mob programming has helped your team focus on the user?

If yes – how?

If no – Why do you think that?

What do you believe is needed to keep a user focus in mob sessions?

Who do you believe has the responsibility of keeping a user focus in mob sessions?