Agile software development and UX design

A case study of integration by mutual adjustment

Persson, John Stouby; Bruun, Anders; Lárusdóttir, Marta Kristín; Nielsen, Peter Axel

Published in:
Information and Software Technology

DOI (link to publication from Publisher):
10.1016/j.infsof.2022.107059

Publication date:
2022

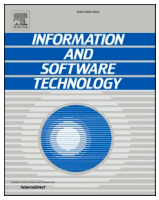
Document Version
Publisher's PDF, also known as Version of record

Information and Software Technology 152 (2022) 107059

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



Agile software development and UX design: A case study of integration by mutual adjustment

John Stouby Persson [a], Anders Bruun [a,*], Marta Kristín Lárusdóttir [b], Peter Axel Nielsen [a]

[a] Department of Computer Science, Aalborg University, Denmark
[b] Department of Computer Science, Reykjavik University, Iceland

A B S T R A C T

Context: Agility is an overarching ideal for empirically-driven software development processes that embrace change in order to improve quality, economy, and simplicity. While the pursuit of Agility has held prominence in software practice and research for over two decades, user experience (UX) designers struggle to integrate their work processes with agile software development.
Objective: As empirical processes are constantly evolving, so is this integration struggle for UX designers. We, therefore, present an industrial case study of how a Danish software company integrates UX design and agile software development.
Method: We conducted a case study involving (a) one iteration of individual interviews with 10 employees (four UX designers, three software developers, two project managers, and one solution architect) and (b) a follow-up iteration consisting of a workshop with 6 employees (three UX designers, two solution architects, and one project manager) two years later. We analyzed how the company's approach to integration with 'upfront design' and 'work in parallel' involve mutual adjustments as opposed to assimilation or separation of UX design and software development.
Results: Our analysis shows how integration through mutual adjustments made distinct contributions to UX designers' and software developers' pursuit of Agility. They experienced notably different work processes that still dealt effectively with change and contributed to quality, economy, or simplicity. Nevertheless, as shown from a follow-up workshop two years after our first interviews, these processes were still susceptible to integration struggles over time.
Conclusion: We conclude that integration based on mutual adjustment potentially makes Agility for UX designers and software developers different and mutually complementary. This integration contrasts with assimilation, which potentially makes their Agility mutually indistinguishably, and with separation, which makes their Agility different and mutually competing.

1. Introduction

Making software systems easy to use has increasingly become a prioritized goal for software development teams over the last three decades. Additionally, giving the users good experiences before, during, and after using software systems has become more acknowledged. Professionals have specialized in this focus and have many different roles, like user experience (UX) designer, UX analyst, UX evaluator, and UX manager [1]. In this paper, we focus on the role of UX designers. To achieve a good user experience, UX designers struggle with influencing software developers; and software developers struggle to stay agile while collaborating with UX designers.

Software developers, for their part, have for more than two decades been influenced by the Agility of software processes. This concern has been explicit since the Agile Manifesto [2] appeared in 2001, but Larman and Basili [3] trace it back to much earlier. According to the state of agile survey, the most popular agile development process is Scrum, with over 80% of participants using that process or some deviations of the process [4]. Other processes mentioned in the survey are Extreme Programming and Lean. Initially, agile development processes were rooted in the software development industry, but lately, agile methodologies are spreading across a broad range of industries [5,6]. However, the substantial literature on agile software development does not provide an unequivocal and standard meaning to the concept of agile processes.

UX designers, who specialize in interaction design and areas different from the particular programming and technical development, struggle to integrate their work into agile processes [7,8]. They may see themselves as "add-ons" to agile development, despite their importance to the success of software projects [9]. User-centered methods and techniques such as comprehensive field investigation and thorough user testing may stand in stark contrast to the quick releases of working code valued in agile development processes. The sometimes conflicting concerns of software developers and UX designers are challenging for integrating their individual efforts for the success of a shared project. To better understand how such integration of work processes is carried out in practice, we present a case study of this integration at a Danish

* Corresponding author.
E-mail address: bruun@cs.aau.dk (A. Bruun).

https://doi.org/10.1016/j.infsof.2022.107059
Received 11 December 2021; Received in revised form 13 June 2022; Accepted 26 August 2022
Available online 27 August 2022

software company to answer the research question:

1.1. How can integrated UX design and software development processes maintain their agility?

With this case study, we report how integrating software development and UX design can rely on mutual adjustment with upfront design and work in parallel. Our analysis, using Conboy's taxonomy of Agility [10], shows the distinct contributions to the software developers' and UX designers' Agility from mutual adjustment. However, these contributions to Agility, revisited in a follow-up workshop in the software company two years later, can dwindle with increased separation. With these insights, our study contributes to the extant research on integrating UX design with software development based on a theoretical and empirical grounded analysis of Agility.

The following Section 2 presents a review of the related literature on agile software development, UX design, and the concepts of Agility and integration. Next, Section 3 presents our case study method, including our choice of the case set in a company with a strong profile in UX design and agile software development and our data collection from 10 individual interviews during the first iteration, a second iteration workshop with 6 participants, and qualitative content analysis. Our findings in Section 4 are on their *upfront design* and *work in parallel*, followed by insights from a workshop for *reflecting on integration*. We finally discuss in Section 5 how our findings contribute to research and practice, followed by a conclusion in Section 6.

2. Related literature

2.1. Agile software development and UX design

The literature on agile software development processes initially had little concern for research on organizational Agility [10,11] and UX design [12]. A theoretical comparison of Scrum and Kanban's fit for UX activities shows that Kanban offers more flexibility and therefore fits better for integrating UX activities into the process [13]. A recent literature review states that the challenges of integrating UX activities into Scrum are related to: the insufficient importance assigned to UX activities in general; insufficient communication between UX designers and developers; insufficient resources assigned to upfront activities in Scrum, and customers trying to represent final users without being aware of their real needs [14]. Another literature review states that the challenges of integrating UX in agile development include lack of time to perform upfront-design and tests with real users, power struggle between UX designers and developers, lack of vision for the whole UX project, difficulty to prioritize UX activities, and lack of documentation [15]. On the contrary, Da Silva et al. [16], state that currently, there is a complete understanding in both communities that UX activities need to be integrated into agile development, and these can not be two separate processes, but more work is still to be done on the tools that address the integration of the agile and UX activities.

One approach to integrating UX activities into agile development is to fit lightweight UX activities into Scrum by using lightweight and incremental processes, conducting simplified think-aloud with pair test users combined with heuristic evaluation [17]. The benefits gained by that integration included more satisfied users; close understanding of users and their needs; better collaboration with stakeholders; and improvements in the development process, including reduced rework. However, neither Scrum nor Kanban supports UX effectively, and they refer to customers significantly more often than to users. Moreover, many processes are tailored to the specifics of development companies, situations, and conditions [9]; for example, having a customer on site in the development room is neglected due to the customer's reluctance to commit the necessary effort or merely the absence of an identifiable customer.

Another approach to tailoring an agile process to the integration of UX activities involves being one iteration ahead by designing the user interface needed for the next iteration [18]. Customer data are gathered up front before the coding starts, design is done one sprint ahead, and user testing is conducted in the next sprint of the coding of that functionality. A systematic review suggests to: (a) conduct little design up front, (b) focus on close collaboration between UX designers and development experts, and (c) make UX designers work one sprint ahead of developers [19]. However, balancing the amount of upfront work and synchronizing between UX designers and software developers is one of the main challenges for their integration [20]. The lack of focus on usability and UX activities in agile software development has attracted growing research interest [8,12,21]. UX designers may have a broad set of responsibilities in agile projects, including sales and business development, which rely on a customer focus rather than a user-centered focus [1]. The UX designers, among other things, struggle with making low-fidelity prototypes at the start of the project and gathering feedback on those, getting collaboration from developers for creating the design, having too short time within each iteration to exploit different design options, and too little time to conduct an evaluation with real users, [22]. Similarly, a lack of collaboration and communication between UX designers and developers was noted by Kuusinen et al. [23], and a lack of understanding of the business and customer needs. A case study documents how UX designers constantly need to justify UX activities and employ salesmanship at the same time to have customers pay for the UX activities [1]. A more recent approach is to integrate Lean UX into agile [24–26]. The Lean process complements the disciplines of agile and UX design with the approach of validated learning using the Lean UX cycle of build, measure, and learn. They suggest that user tests are conducted each week to learn from the feedback gathered from users [24]. Zorzetti et al. [26] conclude that adopting an approach of LeanUX can bring changes in mindsets, activities, practices, and techniques focusing more on the users. The teams in the case studies from that paper recognized that using agile methods alone did not identify whether the right product for users was built but going through the build-measure-learn cycle added the needed understanding of the users' and customers' goals and needs.

A review by Adeola Wale-Kolade et al. [12] identified seven claims across the research literature regarding integrating software development and UX design:

1 Conduct some upfront design activities before project start.
2 Design low-fi prototypes as the basis for developing the system.
3 Perform testing between iterations.
4 Designers and developers work in parallel.
5 Usability designers should be present.
6 Usability designers should be fully integrated into the development team.
7 End users or their proxies should be involved in the project life cycle

These claims have not escaped criticism. For example, claim 5 should ensure UX designers are present, but with only a simple presence, the role may be filled with other personnel. While claim 6 should ensure that UX design concerns are always present through assimilating UX designers into the development team, UX designers may well identify too closely with the software goals and lose track of UX activities and concerns [27]. There are many other rebuttals to these seven claims, showing that the integration of UX activities with agile development is not simple and that matching several concerns is necessary [12].

2.2. Theoretical framing of agility and integration

Researchers and practitioners ascribing the Agility of a software project to the use of a particular process such as Scrum or Kanban may exacerbate the difficulty of integrating software development and UX design. A team may use a process labeled "Agile" in a way that does not create flexibility, leanness, and adaptability [10]. We see Agility as

referring to software developers' and UX designers' effectiveness in dealing with the inevitable changes that affect the success of a software project. Here, we apply a general understanding of Agility, not tied to a specific process, based on Kieran Conboy's systematic literature review of the concept of Agility across different disciplines [10]. Conboy proposes the following definition of Agility, which emphasizes the core principles of embracing change and providing customer value: "(The) continual readiness … to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity) through its collective components and relationships with its environment" [10]. Conboy translated the definition into a formative taxonomy of Agility, stating that to be agile, a process component must:

1 contribute to at least one of the following: (i) creation of change, (ii) proaction in advance of change, (iii) reaction to change, (iv) learning from change;
2 contribute to at least one of and not detract from the following: (i)perceived economy, (ii) perceived quality, (iii) perceived simplicity;
3 be continually ready, i.e., must require minimal time and cost to prepare the component for use.

We use this definition to investigate how one Danish software company that prominently pursues both UX design and agile software development integrates UX design activities with its software development process. While previous research has used Conboy's theory to distinguish contributions to Agility from project- and firm-level processes [11], we use it here to distinguish between contributions to Agility for UX designers and software developers while still needing to integrate their work.

We define *integration* as a distinct way of coordinating activities in organizational processes. Mintzberg's [28] seminal theory of organizations distinguish five prime coordinating mechanisms: *direct supervision, standardization of work processes, standardization of skills, standardization of outputs,* and *mutual adjustment*. The last-mentioned coordinating mechanism is characteristic of organizations with the structure he calls the *Adhocracy*, often situated in complex and dynamic environments [28]. A well-known situation for software companies striving for Agility. The *Adhocracy* and coordinating through mutual adjustment rely on applying diverse and sophisticated expertise as bases for building new knowledge for innovation. Succesful mutual adjustments are the result of informal communication between people conducting interdependent work.

We see mutual adjustment as central to the *integration* of software development and UX design. For further clarity, we distinguish *integration* from *assimilation,* which relies on coordinating by direct supervision and standardization of work processes. Moreover, we see *integration* as different from *separation,* depending on coordination by standardization of skills and outputs, as illustrated in Fig. 1.

Assimilation of agile development and UX design into a process unity, with a single team acting in its own power, has bearings to agile processes rejecting specialized roles to strive for oneness. Here, the coordination of work involves standardized work processes, shared norms of what is desirable, and direct supervision. In contrast to *assimilation, separation* refers to software development and UX design as acting on one another. This twoness has bearings to pre-agile relay races involving the coordination of specialized roles with standardized skills and outputs, e.g., through user studies. *Integration* lies between *assimilation* and *separation*, and it is an across-entities point of view [29] to avoid the reduction into a single entity or the separation into two distinct entities. This *integration* is coordinated by mutual adjustments of the work in the situation at hand. Fig. 1 thus contrasts Mintzberg's [28] coordinating mechanism of mutual adjustment (*integration*) against the four others; two we associate with a single-entity view (*assimilation*) and two with a between-entities view (*separation*). We apply this understanding of *integration* and combine it with Conboy's [10] definition of *Agility* to unfold the relations and social processes [29] of UX design and software development.

3. Method

Our investigation of integrated UX design and software development processes is based on a single case study approach [30,31] to address how these two processes maintain their Agility. A single case study is well suited for studying a contemporary phenomenon in its real-world context [32] to develop insights into our "how" question [33]. This case study design also applies well when the boundary between the phenomenon and the context is unclear, i.e., between the UX design and software development activities and the broader organizational context allowing their Agility. The information-based selection of a single case [32] has the rationale of an unusually [33] high dedication to UX design and agile software development.

3.1. The case

The case setting is a Danish software company, Mjølner Informatics, with 100+ employees who prominently pursue both UX design and agile software development. Mjølner Informatics as a software house develops specialized software products for particular customers on project contracts. Most employees have a master's degree related to UX, interaction design, or software development. The customers are small and large private companies and public organizations. Mjølner Informatics employees may assist customer organizations with specialist knowledge on time-and-materials contracts for up to several months. The Danish edition of *Computerworld* named the company "IT comet of the year, 2015″ in Denmark, partly because of its heavy emphasis on UX in software development. UX designers that play crucial and managerial roles in most projects, account for 10% of the employees.

3.2. Data collection

We collected data in two iterations. First, we conducted 10 individual interviews. We interviewed four UX designers, three software

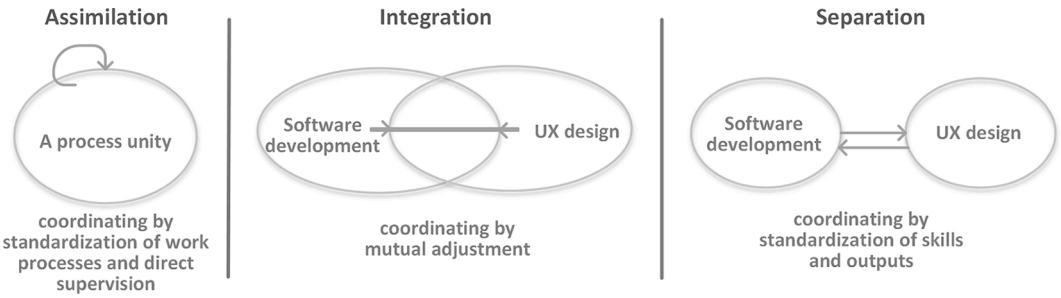


Fig. 1. The three views on the Agility of software development and UX design.

developers, two project managers, and one software architect. Each interview lasted 45–60 min, were audio recorded, and followed Patton's guidelines for pragmatic interviews that *aimed at getting straightforward answers that can yield practical and useful insights* [34]. We had one interviewer and one observer from the research team during each interview. This iteration had a threefold aim of understanding: (1) the agile development process at the company, (2) perceptions of the UX designers' role in different project phases, and (3) the approach of integrating UX design with agile software development.

The second iteration of data collection was conducted two years after the initial interviews. We presented our preliminary findings from the first iteration in a short paper that we shared with the case company during a reflection session with 6 representatives from the case company; three UX designers, two software architects, and one project manager. The project manager also participated in the first iteration two years earlier, while the other participants were new to the study. The software architects also represented the views of software developers as this is the typical career path with an increase in seniority within the company. The session had a two-fold aim of 1) discussing and corroborating our analysis on the integration approach that *was* and 2) discussing the integration approach that *is* two years later to elicit reflections based on our findings presented in Sections 4.1 and 4.2 below. We invited the representatives to reflect on their practices [35], combining our initial interpretation with intervention to better understand the case in its organizational context [36]. Two from the research team facilitated the workshop activity by first providing a recap of our findings from the first iteration of data collection, followed by a discussion. The workshop had a duration of 2 h and was audio recorded.

3.3. Data analysis

We analyzed the data from the 10 individual interviews of the first iteration and the reflection session with 6 participants in the second iteration through a theory-directed qualitative content analysis [37] using Conboy's theory of Agility [10]. Audio recordings from the 10 individual interviews were transcribed in the online tool Dedoose, which we also used to support the content analysis. Two members of the research team conducted the analysis with a particular emphasis on how the effect of integration approaches differs for software developers and UX designers. Audio recordings and notes from the reflection session of the second iteration were analyzed abductively to uncover the participants' views on the integration approach there *was*, how it *is* (two years later), and how they *want* the approach in the future. In Mjølner Informatics, two process elements are essential to integrating UX design with software development in an agile manner: upfront design and work in parallel, which corresponds to the first claim on integrating UX design with an agile development process from Adeola Wale-Kolade et al. [12]: "*Conduct some upfront design activities before project start*" and fourth claim: "*Designers and developers work in parallel*". We present these two in Sections 4.1 and 4.2 and the findings from their workshop reflections on these two in Section 4.3.

4. Findings

4.1. Upfront design

In the upfront design at Mjølner Informatics, the software product is considered from various perspectives before producing any code. Here, UX designers typically participate full-time with responsibilities of defining system requirements, making design sketches, and designing wireframes.

A UX designer in Mjølner Informatics is the main actor collaborating with the customers, the users, and the system architect (an experienced software developer). The system architect and the UX designer collaborate on the validation of wireframes. They consider the system requirements linked to wireframes, which is the primary outcome of the upfront design. One UX designer argues for this upfront design approach by contrasting it with placing UX design one sprint ahead of software development as follows:

> *It is incredibly challenging to create wireframes once development has started. As a UX designer, I then need to work one sprint ahead. I have experienced designing wireframes during a sprint, which is very hectic. The results are not good. The analytic part disappears when these activities are done in the sprint. I lose track and overview. I believe that there is a need to work in an upfront manner, as we do.* (UX Designer 1)

The same UX designer further explains that the one-sprint-ahead approach can work in some cases, but it is complicated and hard to work that way since all sorts of things happen during the sprint. Smaller details can be designed during software development, but an overview through designing wireframes of the whole system needs to be in place during the upfront design period.

The goal of such upfront design is also relevant to software developers. Another UX designer notes that she makes the design to a level of completeness that allows the developers to take over. She explains:

> *The overall design could be a graphical design, but also flows where you, for instance, have identified individual elements, which you then specify further in Jira.* (UX Designer 2)

When software developers take over, they are aware of the challenges and necessity of UX, but some have limited knowledge of the business aspects and the customer's needs. Upfront design is a challenge because the customer often wants to know the "magical price" of the project, and the software developers need to know more about the business aspects. This need forces UX designers to provide specific details up front through UX design, which they can discuss with the customer. A software developer explains:

> *At the beginning of the process, there is this "black box" known as UX design, which is typically positioned before the "tech" phase, when developers enter the project.* (Developer 1)

Software developers do not perceive their limited upfront involvement and knowledge of what is going on in the project as problematic. Instead, a developer states that he appreciates the defined boundaries of UX designers working up front and developers entering the project later:

> *I do not know that much about the very first phase, but it is when you define the overall UX, initial research activities, and initial designs. It happens before developers enter the project… It is nice to join the project when the upfront work has been done; there are some defined boundaries.* (Developer 2)

However, these defined boundaries do not remove the need for making changes. The same software developer explains that when they get more into development, some tasks that were identified up front no longer fit and have to be changed. He wants to keep the focus on upfront design with detailed wireframes and prototypes. A project manager similarly emphasizes the need for continuous adjustments to a product throughout a project. Adjustments to a product made in one sprint could be made in the next sprint or later in the project. But the initial overview in the upfront design period provides a good idea of what to do, he comments.

As shown from the quotes above, upfront design is not used to standardize and rigidly control the design. Instead, both UX designers and software developers use upfront design to achieve Agility, as summarized in Table 1.

Upfront design helps software developers be *proactive* about changes in the system's architectural design by anticipating development risks (see first point's row in Table 1). This is possible thanks to coordination with UX designers, who readily *react* to specification changes from the very beginning and *learn* from these changes, e.g., through heavy use of design iterations in dialog with customers and users. Upfront design activities support initial reactions to change through a process in which

Table 1
Contributions to Agility with upfront design (based on Conboy's [10] framework in the left column).

Agility	Software development	UX design
1. A process component must contribute to at least one of the following: *(i) creation of change* *(ii) proaction in advance of change* *(iii) reaction to change* *(iv) learning from change*	Before development, the system architect is *proactive about changes* and anticipates development risks.	The initial *reaction to change* and *learning from change* are strong with intensive customer dialog and no software committed.
2. A process component must contribute to at least one of and not detract from the following: *(i) perceived economy* *(ii) perceived quality* *(iii) perceived simplicity*	Improved *simplicity* through separation of concerns and less coordination with users.	*Perceived quality* through intensive communication with users and customers and a modest cost of early changes contribute to the *perceived economy*.
3. A process component must be continually ready, i.e., require minimal time and cost to prepare the component for use.	A system architect is available and active throughout the project.	The UX designer autonomously controls the project scope.

early visualizations of ideas through, e.g., paper prototyping, facilitate specific discussions on system requirements before committing to any program code or model implementations. UX designers' responsibility to identify requirements by coordinating with customers and users offers software developers *simplicity* with fewer coordination needs. Also, UX designers improve product *quality* through intensive communication with customers and users. At this early stage, changes in requirements and designs incur modest costs, which contribute to the *perceived economy* (defined as the utilization of all resources is maximized, and no unnecessary resources are maintained [10]). Finally, these achievements of Agility are readily available to both software developers and UX designers, mainly through the activities of the software architect and the UX designer (see the third point's row in Table 1).

Thus, upfront design is an example of integration through mutual adjustment between UX design and software development. This also transcends down to the level of specific tools where UX designers also have editing rights in Jira. It contributes to Agility across the two specialized activities without collapsing them into a single activity or process. The contribution to Agility differs for UX designers and software developers at Mjølner Informatics. As an example, the second row in Table 1 shows that the process followed by the software developers (represented by the architect during upfront design) provides simplicity through a separation of concerns as they have less coordination with users. Such coordination is more intensive in the UX design process, which in turn increases perceived quality.

4.2. Work in parallel

The work in parallel approach in Mjølner Informatics involves software developers producing code based on the specifications elicited in the upfront design and UX designers spending about 20% of their time on a project while also working on other projects. The work in parallel period starts when a development team is formed. The team typically includes one UX designer, two to four developers, and one project manager working in a series of iterations, each lasting three to four weeks. The main responsibility of the UX designer during the work in parallel period is to review and sign off implemented designs at the end of each iteration and facilitate product evaluations with customers. The UX designer is physically located close to the software developers for easy access. The UX designer uses the wireframes initially designed in the upfront design when meeting with the rest of the development team to convey system requirements. A project manager describes this work-in-parallel approach and argues for its necessity for efficient work as follows:

> *At the beginning of the development stage, the infrastructure is set up: Jira and a Scrum board, wireframes, graphical elements, user stories. It is then up to the team to make the necessary refinements … If you have two sprints* n *and* $n+1$*, the UX designer is central in sprint* n *to prepare sprint* $n+1$*. They need to be ahead so that software developers have some designs to work with; otherwise, it will not be efficient work.* (Project Manager 1)

A UX designer continues the project manager's line of reasoning as an aim of her design practice. She explains that while being in one sprint, she needs to begin working on designs for the next sprints. Typically, UX designers would aim to begin one or two sprints ahead. A software developer elaborates on UX designers' responsibilities:

> *Sometimes the UX designer assumes a reviewing role to go over our implemented designs, and this is the case throughout the project.* (Developer 2)

The UX designer acknowledges the role's wide range of responsibilities, which implies limited prerequisite knowledge, making frequent informal communication with developers necessary. She stresses that the development of designs must never become top-down, and there will be some elements that she cannot know. She further explains:

> *So, we need to engage during development … I spend about one-sixth of my time during development in projects. During development, we participate in the planning to answer questions from the developers.* (UX Designer 2)

A project manager further elaborates UX designers' work in parallel approach. He says that UX designers typically engage full-time in the upfront design of the project, and then their work is phased out. He also comments that daily collaboration with software developers is essential since there will be some minute changes in the design until the final semicolon is set in the code. He further explains:

> *This works well at Mjølner since the architect, project manager, and UX designer are in the same building or room, so this comes naturally, also as part of the daily Scrum meetings … The UX designer is part of the team to ensure that requirements are met. Some would think that UX designers create lovely designs and icons, but this is not the central part of that role.* (Project Manager 1)

As shown from the quotes above, the work in parallel approach is not aimed at a rigid division of labor or at defending a level of control for particular roles. Instead, both software developers and UX designers use the work in parallel approach to achieve Agility, as summarized in Table 2.

The work-in-parallel approach involves a UX designer meeting with the software developers at the end of each sprint to review implemented designs. This limited time scope of the reviews makes *reactions to change* more manageable for software developers, as they do not have to continuously cope with changes (see first point row in Table 2). On the other hand, this enables the UX designer to focus on *creating change* from the customer perspective. The structured UX input through e.g. specific visualizations of requirements contributes to software developers' *perceived simplicity* and localized efforts at specific potnts during work-in parallel contributes to UX designers' *perceived economy* of scale as this enables them to work on multiple projects. Finally, these achievements of Agility are readily available to both software developers and UX designers (see the third point's row in Table 2). Overall, work in parallel is integration through mutual adjustment between UX design and software development, and it contributes to Agility in different ways for UX designers and software developers.

Table 2
Contributions to Agility with work-in-parallel.

Agility	Software development	UX design
1. A process component must contribute to at least one of the following: *(i) creation of change (ii) proaction in advance of change (iii) reaction to change (iv) learning from change*	UX feedback and design changes limited to sprint reviews make *reactions to change* more manageable.	Thorough reviews and quality control detached from the development team while in dialog with customers promote the *creation of change*.
2. A process component must contribute to at least one of and not detract from one of the following: *(i) perceived economy (ii) perceived quality (iii) perceived simplicity*	*Perceived simplicity* through structured UXdesign inputs that visualize requirements using e.g. wireframes.	*Perceived economy* of scale through localized efforts during work-in-parallel enables the ability to work on multiple projects.
3. A process component must be continually ready, i.e., require minimal time and cost to prepare the component for use.	Readily available designs and designers.	Reuse of UX competencies and insights across projects.

In Mjølner Informatics, the two approaches upfront-design and work-in-parallel to integrating UX design with software development contribute to Agility. However, software developers' and UX designers' experiences differ in their dealings with change, simplicity, quality, economy, and readiness.

- Software developers use upfront design to proact in advance of change for the sake of simplicity, whereas UX designers use upfront design to react and learn from change for the sake of quality and economy (Table 1).
- Software developers use work in parallel to respond to change for the sake of simplicity, whereas UX designers use work in parallel to create change for the benefit of an economy of scale (Table 2).

4.3. Reflecting on integration

Our final phase of inquiry revolved around presenting our findings on the integration approach in terms of upfront design and work-in-parallel. We wanted to provide feedback to the case company related to our interpretations using Conboy's theory on Agility and Mintzberg's coordination mechanisms as the analytical lens on their integration. As highlighted in the previous sections, we discovered upfront design and work-in-parallel phases in which software developers and UX designers coordinated through mutual adjustment, representing an approach resembling integration rather than assimilation and separation. We asked the participants to reflect on Mjølner's approach to encompassing agile software development and UX design through an integration approach. We asked them to reflect on how this approach *was* and the extent to which they could recognize it through our analysis. We also asked participants to reflect on how their agile integration approach currently *is* and how they *want* it to look like in the future. Through this reflection, we identified three main points.

4.3.1. Agile integration considered better the way it was

During our inquiry on reflection, it became apparent that Mjølner transitioned into an approach resembling that of separation. One of the architects stated that he was now much less involved in upfront design than three years ago. In his view, the upfront design phase now seemed overly dominated by UX designers.

> *"Things have happened over the past few years. Customers are no longer willing to pay for upfront activities to the same extent. This means that we now must send only one person, and other times we do not even send one out because the project is defined by the customer already. In that situation it is difficult to say when the upfront design starts because we are not part of the initial exploratory phase. We need to go back to the situation where the UX designer and architect had a high level of collaboration. I want to be able to push back to the UX designer and* vice versa *to ensure the best solution is developed… I would really like to be part of the process rather than having to interpret others' findings through artifacts in Jira."* (Architect 2)

This architect has the impression that there needs to be a more even balance between the UX designers' and architects' responsibilities during up-front design. He found it critical that these roles mutually adjust during this phase for enabling architects to proactively react to change on different technical issues and risks, which are problems outside the typical expertise area and scope of UX designers. In reasoning why the current integration approach is that way, the architect mentioned that the company is experiencing a shift in the market where customers now find it too expensive to assign multiple people to conduct exploratory upfront design activities. In the following, we deal with this point in more detail.

4.3.2. Agile integration depends on the project and customer

Although Mjølner has transitioned into an approach resembling separation, the architects, software developers, and UX designers in some projects integrate by coordinating through mutual adjustments. In such cases, a software architect is more involved during upfront design together with a UX designer. One of the architects stated that this particularly applies when developing software to be used in embedded products.

Additionally, there seems to have been a shift in the role of upfront design due to changing customer needs. This change is closely tied to the point of Mjølner having transitioned into the separation approach. One mentions that as an architect, he is missing the activities that go into fundamentally understanding the domain in which developed systems are supposed to operate. Customers are less willing to pay for such upfront activities. Now, customers are eliciting requirements and designs on their own, on which basis Mjølner is now just supposed to develop a solution. This change is corroborated by one of the UX managers:

> *"I need to highlight that we are now operating in a different market. Generally speaking, many companies now need an IT department no matter what type of company they are. These IT departments want to solve several of the upfront design issues on their own. This also means that these companies, our customers, now want a higher degree of project ownership on areas that we were responsible for in the past."* (UX Manager 2)

As illustrated in the quote above, one of the UX managers now saw a trend for companies to create their own IT departments that will take many exploratory upfront design tasks, which Mjølner was formerly hired to do. This trend influences Mjølner in terms of their UX professionals' ability to initially react to changes and learn from these (first point's row in Table 1), given that the impact through, e.g., customer dialog is reduced with increased fixation of requirements.

4.3.3. Want to transition from separation towards integration

Based on the above observations, Mjølner has transitioned from an integration approach to separation in which software developers and UX designers are now trying to coordinate through standardization of skills and output. One of the UX managers mentioned that the UX designers want the projects to be less person-dependent.:

> *"I believe, on behalf of the UX designers, that we have mostly worked in an individual manner to create good relations between ourselves and*

architects. It is probably time for us to be less dependent on individuals. It should not be about us as UX designers to tell others what we do and what should be done. Rather, it should be about UX designers sitting down with architects and developers to figure out how to collaborate. It's crucial to have a stable set of fixpoints we can orient ourselves towards." (UX Manager 2)

One of the software architects was also explicit on behalf of the software developers in stating that the current separation approach is flawed since the output is not standardized, i.e., the tools used by UX designers during a project are highly dependent on the persons participating. As an alternative, the architect would like to see more widespread use of abstract design tools such as use case diagrams that software developers can relate to. These shared artifacts and design languages enabled coordination through mutual adjustment in the past.

"Some years ago, we had a shared toolbox between UX designers and architects that everyone were able to apply. We had set of shared artifacts and methods that could be used. Currently, we do not have this shared toolbox. The set of artifacts and methods used depends heavily on the persons involved. We want to go back to the past situation where no matter which people we put on projects, they were able to collaborate and knew exactly how to use the tools and methods." (Architect 2)

In summary, Mjølner *is* currently following an approach of separation, albeit with some challenges in standardizing skills and outputs, but they *want* to transition back into the how it *was*, i.e. integration. They perceived a need for re-obtaining a shared language between UX designers and software developers. Table 3 summarizes our findings from the reflection session in relation to Agility.

In Table 3, the first point's row highlights how the software architects are now less involved during upfront design activities. The architects and UX designers have drifted into a more reactive role with changes in their customer market. This change has shifted the architects and UX designers into a position of reacting to customers' specifications compared to the previously more collaborative and creative nature of their roles two years prior. The second point's row of Table 3 shows how simplicity and economy are now perceived differently than two years earlier, which in the way of working is perceived by the architects and UX designers as potentially having negative consequences on quality. The third point's row in Table 3 outlines how the process components in the current situation are less readily available, partly due to not being involved during upfront design activities, but in particular also through the challenge of individual dependencies and not having a shared design language and toolbox. Overall, Table 3 outlines how their Agility has become more different and separated, making their practices mutually compete for Agility. This situation is distinctively different from the findings from two years earlier (c.f. Tables 1 and 2), where they integrate their activities by mutual adjustment. Moreover, it shows that integrating by mutual adjustments is itself susceptible to change and that reflecting on integration helps revisit their state of agile and point out opportunities for changing their situation.

Table 3
Contributions to Agility *as is* in relation to upfront design and work-in-parallel.

Agility	Software development	UX design
1. A process component must contribute to at least one of the following: *(i) creation of change* *(ii) proaction in advance of change* *(iii) reaction to change* *(iv) learning from change*	**Upfront Design** Involved in some projects, but not all. Challenging to be *proactive*, no shared design language between software developers and UX designers. **Work in Parallel** Drift from *reacting to changes* through collaboration with UX design towards reacting to handovers from customers.	**Upfront Design** *Reactive* to changes and *learning from change*, but not in all projects. Tool use depends on the individual. **Work in Parallel** Drift from a role as *creator of change* to more simple *reactions to changes* specified by customers.
2. A process component must contribute to at least one of and not detract from one of the following: *(i) perceived economy* *(ii) perceived quality* *(iii) perceived simplicity*	**Upfront Design** *Perceived economy* due to not participating in upfront design activities comes at the cost of *perceived quality*. **Work in Parallel** *Perceived simplicity* through pre-made specifications from customers and UX designers comes at the cost of *perceived quality* and *economy* because some specifications are challenging to implement, but also because UX output does not provide a shared design language.	**Upfront Design** *Perceived simplicity* through not having to coordinate with an architect. In some cases, the customer conduct upfront design, which contribute to *perceived simplicity* and *economy*, but at the cost of *perceived quality*. **Work in Parallel** *Perceived simplicity* through pre-made specifications from customers (in some projects) comes at the cost of *perceived economy* and *quality* by implementation challenges.
3. A process component must be continually ready, i.e., require minimal time and cost to prepare the component for use.	**Upfront Design** Software developers and system architects are not always present during upfront design activities. **Work in Parallel** Readily available developers and designers. Designs are not based on a shared design language nor based on mutual adjustments and are therefore not readily available.	**Upfront Design** The UX designer is readily available, not in all projects, however, due to customer autonomy in creating specifications. **Work in Parallel** UX competencies and insight may not necessarily be reused across projects due to not having a shared design language.

5. Discussion

Our study aimed to address the research question: *How can integrated UX design and software development processes maintain their Agility?* Hence, we presented a case study of the integrated UX design and software development processes at the Danish software company Mjølner Informatics. This section discusses how our study contributes to the extant research, its implications for practice, limitations, and future research.

5.1. Contributions to research

Two concepts were central to our case study of agile software development and UX design at Mjølner Informatics. The first was *Agility*, for which we used Conboy's [10] taxonomy that allowed us to be open to the meaning of Agility in our specific case rather than judging it according to some of the many methodologies available in the literature (e.g., [13,14,20,21]). The second was *integration*, which we defined as involving mutual adjustments based on Mintzberg's classical theory of organizations [28] and thereby differentiated *integration* from other ways of coordinating through *standardization of work processes, skills, and outputs,* or *direct supervision*. From analyzing our case, we show how *Agility* may differ for UX designers and software developers when integrating their efforts in a software project through upfront design (cf. Section 4.1) and work in parallel (cf. Section 4.2). This finding contributes to previous research on the challenges of integrating UX into agile processes [14] by showing how the ideals of successful integration can be specific to the role and situation.

Next, we showed in our workshop (cf. Section 4.3) that these concepts also were helpful to the practitioners for reflecting on what *was*, presently *is*, and what they *want* for their integration. The conceptualizations provided a starting point for deliberation on their practices in the workshop. While this starting point for deliberation is more abstract by

focusing on the overall integration process than previous research on, e.g., concise user stories [8] or technical debt items [38], it still allows positioning and discussing *Agility* in their specific situation. We use the three views on the Agility of software development and UX design (cf. Fig. 1) to explain our case study at Mjølner Informatics and as a basis for proposing the three distinct claims about Agility presented in Fig. 2.

On the left in Fig. 2, we have *assimilation* that involves coordinating by standardization of work processes and direct supervision, which potentially makes Agility for UX and software to be mutually indistinguishable. Remnants of this view were present in our case, as UX has been and still is a concern for the software developers and goes back to agile methods such as Scrum [39] and extreme programming [40] that were skeptical of technical roles beyond that of a team member.

The *integration* view that involves coordinating by mutual adjustment, which potentially makes Agility for UX and software to be different and mutually complementary, was a central focus in our case study. Our findings from Mjølner Informatics' (cf. Sections 4.1 and 4.2) substantiated this claim in practice. However, as shown in a workshop two years after our first interviews, this *integration* is fragile and may drift towards *separation,* as shown on the right in Fig. 2.

The *separation* view, which involves coordinating by standardization of skills and outputs, potentially makes Agility for UX and software to be different and mutually competing. Our case study found that changes in Mjølner Informatics' market (what Mintzberg calls a situational factor [28]) further pushed the software developers and UX designers towards *separation*. This market orientation shifted some of the UX design activities into the customer organization, which made it difficult for Mjølner Informatics' internal UX designers to adjust to the concerns of the software developers. The software developers disliked this *separation* from UX and requested they return to integration at our workshop. However, a single case study like ours cannot claim that any of the three is superior to the two others, only that they are feasible and that UX design and software development processes may transition between them over time.

Overall, Fig. 2 distinguishes three relationship types based on Mintzberg's theory of organizations [28] as having inherent views on Agility. To unfold these views, Tables 1 and 2 (cf Sections 4.1 and 4.2) are exemplars for analyzing the specific contributions to Agility in a software development and UX design relationship on a more detailed level. Table 3 (cf. Section 4.3) further shows how Conboy's [10] taxonomy can be useful for reflecting on maintaining Agility according to the three claims in Fig. 2.

5.2. Implications for practice

Our case study has some practical implications, and we propose a three-step inquiry to help practitioners understand integration in a specific situation or project to realize these implications. The first step is to identify approaches to integrating UX design with software development. Here, a project manager or someone dedicated to facilitating Agility, such as a Scrum master, may prefer assimilation or separation over integration (see Fig. 2). A preference for assimilation may be rooted in the team-centric principles in the agile manifesto [2] or Scrum guide [41] for the all-inclusive label of developers to be collectively accountable for their work. In contrast, a preference for separation may be rooted in the distinctions of roles and tasks from frameworks such as the Unified Process [42]. The project manager or Scrum master can identify approaches to integration using the previously discussed seven claims regarding agility and UX design [12]. However, these seven claims are not an exhaustive list; other approaches may be more specific to the situation. The second step is to assess the contributions to Agility for both software development and UX design. This assessment considers a process component's contribution to managing change, perceived economy, quality, and simplicity, and its continual readiness for use [10]. Tables 1–3 present examples of such an assessment. The third step involves reflecting on their mutual adjustments inherent to these assessments of the situation in order to improve Agility. Our study shows (cf. the workshop presented in Section 4.3) how this reflection is useful for uncovering an unwanted drift toward separation; thus, we believe that revisiting these three steps also can help proactively avoid it.

5.3. Limitations

Our single case study of UX design and software development processes provides evidence of what is feasible for their integration and Agility. We can not make any claims on what is preferable or effective on a generalized level. Unfolding the concepts of Agility and Integration with detailed empirical insights from a single case contributes to richness rather than representation. Our case was not chosen to be representative of agile software development and UX design integration in most cases but as an unusual case [32] with a Danish organization highly dedicated to both UX and Agility. This information-oriented case selection for integration of Agility and UX Design implies analytical rather than a statistical generalization. Our case study's analytic generalization advances theoretical concepts [33], specifically the theory of Agility, to consider the practices of upfront design and work in parallel as potential contributions to the Agility of UX design and software development. This finding, although limited in terms of statistical generalizability, is interesting because, according to earlier research, these two practices detract from Agility in software development.

5.4. Future research

Our study points to ample opportunities for future research that compares the effect on Agility from different approaches to integration. Other researchers may conduct comparative case studies or surveys of multiple organizations to determine such effects. Our findings from an unusual case set in an organization highly dedicated to both UX and Agility may be empirically tested by comparison with a representative

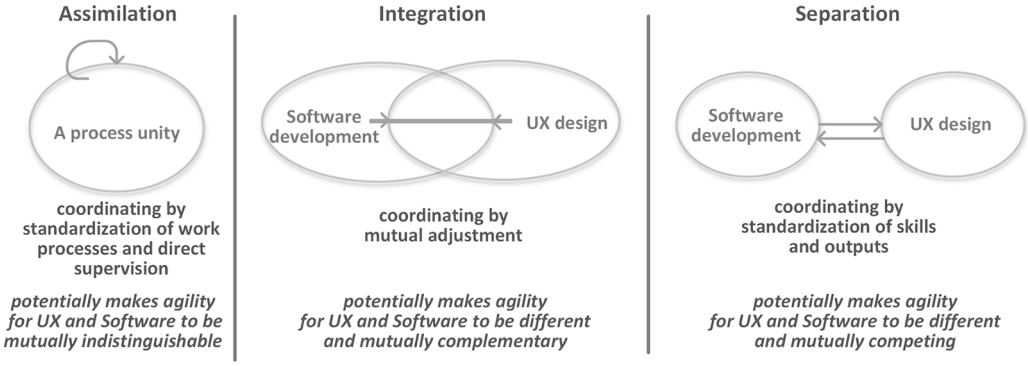


Fig. 2. Three claims about Agility for software development and UX design.

sample of organizations' integration practices. Moreover, we found that situational factors [28], such as the organization's market environment, may influence the coordination and integration of UX design and software development processes. For organizations, situational factors may include their age and size, technical systems, environment, and power structures [28], which could be important for explaining how they integrate UX design and software development processes. Finally, future research may also test the usefulness and transferability of our findings through action research, similar to previous efforts on the challenges of integrating UX work with agile software development [8]. A starting point could be to test and theoretically elaborate the three steps outlined in the previous section on implications for practice.

6. Conclusion

To answer how integrated UX design and software development processes can maintain their Agility, we present a case study at Mjølner Informatics', a company highly dedicated to UX design and agile software development. We analyzed the company's integration approaches of upfront design and work in parallel with Conboy's taxonomy of Agility [10]. This analysis showed how Agility differs for the two roles with these two integration approaches. They experienced notably different work processes that still dealt effectively with change and contributed to quality, economy, or simplicity. We explain that their integration through mutual adjustment makes the Agility for UX designers and software developers different yet complementary. This integration contrasts with assimilation, which potentially makes their Agility mutually indistinguishable, and with separation, which makes their Agility different and mutually competing.

Our follow-up workshop two years after our first interviews also showed that the processes of upfront design and work in parallel were susceptible to integration struggles over time. At that point, we found a drift towards separation, making their Agility increasingly different and mutually competing. This finding suggests practitioners should reflect more frequently on how their integration approaches afford Agility and to whom.

CRediT authorship contribution statement

John Stouby Persson: Conceptualization, Methodology, Visualization, Formal analysis, Investigation, Writing – original draft. **Anders Bruun:** Formal analysis, Investigation, Writing – original draft, Project administration. **Marta Kristín Lárusdóttir:** Investigation, Writing – original draft, Project administration. **Peter Axel Nielsen:** Writing – review & editing, Visualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Bruun, M.K. Larusdottir, L. Nielsen, et al., The role of UX professionals in agile development: a case study from industry, in: G. Berget (Ed.), Proceedings of the 10th Nordic Conference on Human-Computer Interaction, ACM, 2018, pp. 352–363.

[2] Beck, K., Beedle, M., Van Bennekum, A. et al.: Manifesto for Agile Software Development. (2001).

[3] C. Larman, V.R. Basili, Iterative and incremental developments. A brief history, Computer 36 (2003) 47–56 (Long Beach Calif).

[4] Digital.ai: The 15th Annual State of Agile Report (https://Digital.Ai/Resources/State-of-Agile). (2021).

[5] S. Balaban, J. Đurašković, Agile project management as an answer to changing environment, Eur. Proj. Manag. J. 11 (2021) 12–19.

[6] D.Ø. Madsen, The evolutionary trajectory of the agile concept viewed from a management fashion perspective, Soc. Sci. 9 (2020) 69.

[7] M. Larusdottir, J. Gulliksen, Å. Cajander, A license to kill–improving UCSD in agile development, J. Syst. Softw. 123 (2017) 214–222.

[8] A. Ananjeva, J.S. Persson, A. Bruun, Integrating UX work with agile development through user stories: an action research study in a small software company, J. Syst. Softw. 170 (2020), 110785.

[9] K. Schmitz, R. Mahapatra, S. Nerur, User engagement in the era of hybrid agile methodology, IEEE Softw. 36 (2018) 32–40.

[10] K. Conboy, Agility from first principles: reconstructing the concept of agility in information systems development, Inf. Syst. Res. 20 (2009) 329–354.

[11] J.S. Persson, J. Nørbjerg, P.A. Nielsen, Improving ISD agility in fast-moving software organizations, Anonymous, in: Proceedings of the 24th European Conference on Information Systems, AIS, İstanbul,Turkey, 2016, pp. 1–16.

[12] A. Wale-Kolade, P.A. Nielsen, T. Päivärinta, Usability work in agile systems development practice: a systematic review. Anonymous Building Sustainable Information Systems, Springer, 2013, pp. 569–582.

[13] E.L. Law, M.K. Lárusdóttir, Whose experience do we care about? Analysis of the fitness of scrum and kanban to user experience, Int. J. Hum. Comput. Interact. 31 (2015) 584–602.

[14] D. Argumanis, A. Moquillaza, F. Paz, Challenges in integrating SCRUM and the user-centered design framework: a systematic review, in: V. Agredo-Delgado, K. O. Villalba-Condori, P.H. Ruiz (Eds.), Iberoamerican Workshop on Human-Computer Interaction, Springer, 2020, pp. 52–62.

[15] K. Curcio, R. Santana, S. Reinehr, et al., Usability in agile software development: a tertiary study, Comput. Stand. Interfaces 64 (2019) 61–77.

[16] T.S. Da Silva, M.S. Silveira, F. Maurer, et al., The evolution of agile UXD, Inf. Softw. Technol. 102 (2018) 1–5.

[17] D. Teka, Y. Dittrich, M. Kifle, Adapting lightweight user-centered design with the scrum-based development process, Anonymous, in: Proceedings of the ACM/IEEE Symposium on Software Engineering in Africa, 2018, pp. 35–42.

[18] D. Sy, Adapting usability investigations for agile user-centered design, J. Usability Stud. 2 (2007) 112–132.

[19] T.S. Da Silva, A. Martin, F. Maurer, et al., User-centered design and agile methods: a systematic review, Anonymous, in: Proceedings of the Agile Conference, IEEE, 2011, pp. 77–86.

[20] M. Brhel, H. Meth, A. Maedche, et al., Exploring principles of user-centered agile software development: a literature review, Inf. Softw. Technol. 61 (2015) 163–181.

[21] D. Salah, R.F. Paige, P. Cairns, A systematic literature review for agile development processes and user centred design integration, Anonymous, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, ACM, 2014, pp. 1–10.

[22] A.P.O. Bertholdo, F. Kon, M.A. Gerosa, Agile usability patterns for user-centered design final stages, in: M. Kurosu (Ed.), Proceedings of the International Conference on Human-Computer Interaction, Springer, 2016, pp. 433–444.

[23] K. Kuusinen, T. Mikkonen, S. Pakarinen, Agile user experience development in a large software organization: good expertise but limited impact, Anonymous, in: Proceedings of the International Conference on Human-Centred Software Engineering, Springer, 2012, pp. 94–111.

[24] J. Pilz, J. Deutschländer, J. Thomaschewski, et al., Integrating agile human-centered design with lean UX and scrum, Anonymous, in: Proceedings of the 17th International Conference on Web Information Systems and Technologies, 2021, pp. 467–473.

[25] I. Signoretti, S. Marczak, L. Salerno, et al., Boosting agile by using user-centered design and lean startup: a case study of the adoption of the combined approach in software development, Anonymous, in: Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE, 2019, pp. 1–6.

[26] M. Zorzetti, I. Signoretti, L. Salerno, et al., Improving agile software development using user-centered design and lean startup, Inf. Softw. Technol. 141 (2022), 106718.

[27] M. Detweiler, Managing UCD within agile projects, Interactions 14 (2007) 40–42.

[28] H. Mintzberg, Mintzberg on Management: Inside our Strange World of Organizations, The Free Press, New York, NY, 1989.

[29] C. Morgner, Reinventing social relations and processes:john dewey and trans-actions, in: C. Morgner (Ed.), John Dewey and the Notion of Trans-action, Springer, 2020, pp. 1–30.

[30] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2009) 131–164.

[31] J. Lazar, J.H. Feng, H. Hochheiser, Research Methods in Human-Computer Interaction, 2nd ed., Morgan Kaufmann, Cambridge, MA, 2017.

[32] B. Flyvbjerg, Five misunderstandings about case-study research, Qual. Inq. 12 (2006) 219–245.

[33] R.K. Yin, Case Study Research and applications: Design and Methods, 6th ed., 5, Sage Publications Inc, 2018.

[34] M.Q. Patton, Qualitative Research & Evaluation methods: Integrating Theory and Practice, Sage Publications, 2015.

[35] D.A. Schön, The Reflective Practitioner: How Professionals Think in Action, Basic Books, New York, NY, 1983.

[36] K. Braa, R. Vidgen, Interpretation, intervention, and reduction in the organizational laboratory: a framework for in-context information system research, Account. Manag. Inf. Technol. 9 (1999) 25–47.

[37] H. Hsieh, S.E. Shannon, Three approaches to qualitative content analysis, Qual. Health Res. 15 (2005) 1277–1288.

[38] N.B. Borup, A.L.J. Christiansen, S.H. Tovgaard, et al., Deliberative technical debt management: an action research study, in: X. Wang, A. Martini, A. Nguyen-Duc, et al. (Eds.), Proceedings of the 12th International Conference on Software Business, Springer, 2021, pp. 50–65.

[39] K. Schwaber, Scrum development process, Anonymous. Business Object Design and Implementation, Springer, 1997, pp. 117–134.
[40] Beck, K.: Extreme programming explained: embrace change. addison-wesley professional (2000).
[41] Schwaber, K., & Sutherland, J.: The scrum guide: the definitive guide to scrum: the rules of the game (2020).
[42] C. Larman, Agile and Iterative Development: A Manager's Guide, Pearson Education Inc., Boston, MA, 2004.