International Journal of Human-Computer Interaction

ISSN: 1044-7318 (Print) 1532-7590 (Online) Journal homepage: https://www.tandfonline.com/loi/hihc20

Whose Experience Do We Care About? Analysis of the Fitness of Scrum and Kanban to User Experience

Effie Lai-Chong Law & Marta Kristín Lárusdóttir

To cite this article: Effie Lai-Chong Law & Marta Kristín Lárusdóttir (2015) Whose Experience Do We Care About? Analysis of the Fitness of Scrum and Kanban to User Experience, International Journal of Human-Computer Interaction, 31:9, 584-602, DOI: 10.1080/10447318.2015.1065693

To link to this article: https://doi.org/10.1080/10447318.2015.1065693

Intl. Journal of Human–Computer Interaction, 31: 584–602, 2015

ISSN: 1044-7318 print / 1532-7590 online
DOI: 10.1080/10447318.2015.1065693

Taylor & Francis
Taylor & Francis Group

Whose Experience Do We Care About? Analysis of the Fitness of Scrum and Kanban to User Experience

Effie Lai-Chong Law[1] and Marta Kristín Lárusdóttir[2]
[1]*Department of Computer Science, University of Leicester, Leicester, United Kingdom*
[2]*School of Computer Science, Reykjavik University, Reykjavik, Iceland*

Two project management approaches, Agile and Lean, have increasingly been adopted in recent years for software development. Meanwhile, in the field of human–computer interaction (HCI), user experience (UX) has become central in research and practice. The new hybrids between the two fields—Agile UX and Lean UX—were born a few years ago. As Agile, Lean, and UX have different principles and practices, one can query whether the couplings are well justified and whether Agile or Lean is more compatible with UX work. We have conducted a conceptual analysis and tended to conclude that Lean instantiated as Kanban fits UX work better than Agile instantiated as Scrum. To explore further our claim, we performed a secondary data analysis of 10 semistructured interviews with practitioners working with Scrum and Kanban in different sectors (Study 1). This study enabled us to gain insights into the applications of the two processes in real-life cases, their strengths and weaknesses, and factors influencing the practicality of implementing them. Both processes seem not favorable for UX work in practice. Among others, one intriguing observation is loose adherence to the related guidelines and principles. A query derived from the analyses of the interviews is that "customer," as compared with "user," has more frequently been referred to by our interviewees, irrespective of the process they adopted. We have then been motivated to investigate this issue, using a web-based survey with another batch of practitioners ($N = 73$) in the software industry (Study 2). Results of the survey indicate that the practitioners in general had a reasonable understanding of the concepts "user" and "customer," although a minority tended to treat them as synonyms. Limitations of the current studies and implications for future work are discussed.

1. INTRODUCTION

At the turn of the millennium, two new approaches—user experience (UX) and Agile—have evolved almost simultaneously in the two closely related fields, namely, human–computer interaction (HCI) and software engineering (SE), respectively. Agile emerged in response to the need for lightweight software development methods to address drawbacks of their heavy plan-driven counterparts. Similarly, the inception of UX was triggered by the limitations of pragmatic performance-based usability methods to address a wider scope of cognitive, affective and emotional responses of users of ever-increasingly diverse interactive systems. In accord with ISO 9241-210:2010, we regard that usability is subsumed by UX (para. 2.15 "user experience"), which in turn is subsumed by UCD (para. 4.6 "the design addresses the whole user experience"; cf. Figure 1).

Forging hybrid research and practice between HCI and SE has been endeavored for decades (e.g., Seffah, Vanderdonckt, & Desmarais, 2009). The application of the User-/Human-Centered Design (UCD or HCD[1]) framework in software development processes has much been encouraged (Hocko, 2011). The emergence of Agile UCD (Beyer, 2010), Agile UX (e.g., Miller & Sy, 2009), and its "spin off" Lean UX (Gothelf, 2013) can be considered as the continuation of such an effort. Similar to UCD, agile software development and lean software development have their own specific set of principles, practices, and tools, which reflect the values and assumptions underpinning the respective work. Although some of the principles and practices are common to the three approaches—UCD, Agile, and Lean (e.g., upholding the goal of delivering user value)—some are unique (e.g., time-boxed constraint as sprints in Scrum; limiting the amount of work in progress in Kanban; addressing the whole user experience in UCD), and some may be even incompatible (e.g., holistic design of UCD versus reductionist slicing of work in Agile and Lean).

Although the two approaches—UX and Agile—have coexisted for more than a decade, a standardized approach to integrating UX work into mainstream agile development

Address correspondence to Effie Lai-Chong Law, Department of Computer Science, University of Leicester, University Road, LE1 7RH Leicester, United Kingdom. E-mail: lcl9@le.ac.uk; or Marta Kristín Lárusdóttir, School of Computer Science, Reykjavik University, Menntavegi 1, 101 Reykjavik, Iceland. E-mail: marta@ru.ist

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/hihc.

[1]In ISO 9241-210:2010, the term "human-centered design" is used instead of "user-centered design" with the underlying rationale of emphasizing that this part of ISO 9241 addresses impacts on stakeholders other than users. Nonetheless, in practice, the two terms are used interchangeably.

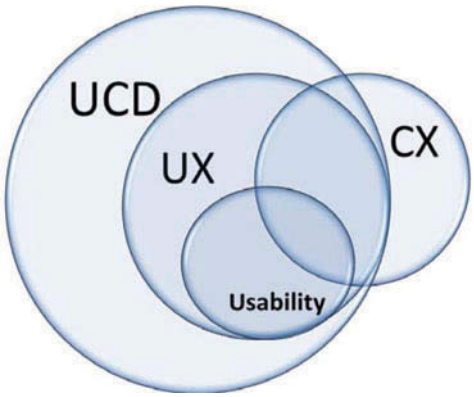

FIG. 1. Relations between related concepts: UCD (user-centered design), UX (user experience), Usability, and CX[2] (Customer Experience). *Note.* We derive the nested UCD-UX-Usability relation from ISO 9241-210:2010. The relation between CX and the others are based on our interpretation of the CX definition (Footnote 2). Although CX is not the focus of this article, its relation with UX should further be studied.

processes such as Scrum and extreme Programming is yet to be established. Avoiding the use of big design upfront, Agile processes aim to accommodate new requirements by developing software iteratively and incrementally in short and fast cycles. Nonetheless, such avoidance is known to have detrimental effects on the coherence of software structure and on the synchronization of different design and development tasks (e.g., Da Silva, Silveira, & Maurer, 2013; Da Silva, Silveira, Maurer, & Hellmann, 2012). Several challenges for UCD–Agile integration and associated practices for tackling them have been identified in a recent systematic literature review (Salah, Paige, & Cairns, 2014). First, a separate predevelopment phase (sprint 0) called "upfront design" is included to realize the advantages of understanding users, tasks, and contexts (i.e., one of the UCD principles). Second, Agile teams support flexible chunking (or time-boxing) of design activities to accommodate the breadth and depth of UX activities. Third, communicating design vision as early and frequently as possible to optimize the dynamics of workflow between developers and UX practitioners. Fourth, discount usability tests with certain compromising strategies (e.g., a limited number of users, UX specialists as surrogate users or user interface inspectors, and informal wiki-based documentation) are adopted to fit into the tight Agile development schedule. Although the proposed practices appear promising for integrating Agile and UX, two issues need to be further examined: to what extent these practices are realized in software industry, and whether as well as how these practices can be adapted for integrating Lean with UX. Considering the different principles and practices, there is a basic question to be explored: *Which of the two—Agile instantiated as Scrum or Lean instantiated as Kanban—can better support UX research and practice?*

There have been some attempts to study the integration of Scrum and UX (e.g., Dingsøyr, Nerur, Balijepally, & Moe, 2012; Kuusinen, Mikkonen, & Pakarinen, 2012; Lárusdóttir, Haraldsdottir, & Mikkelsen, 2009), although the uptake of the proposed methods such as *Design Studio* (Ungar & White, 2008) or *Office Hours* (Leszek & Courage, 2008) by the industry remains unclear. However, to our best knowledge, the integration of UX and Kanban, as a newer approach, remains largely unexplored in the related literature. We aim to investigate the issue by comparing the relative fitness of Scrum and Kanban to UX work, first analytically and then empirically. Specifically, we first describe research and practices pertaining to UX, Scrum and Kanban, which are the two commonly used processes of Agile and Lean (Diebold & Dahlem, 2014), respectively. We then present a conceptual analysis of comparing Scrum and Kanban characteristics, as well as assessing their compatibility with UX work. Furthermore, we have performed a secondary data analysis of our interviews with 10 practitioners from software industry to gain insights into the actual applications of the two processes in business cases, their respective strengths and weaknesses, and factors influencing their practicality.

In addition, we analyzed how collaborations among the stakeholders were coordinated and implemented in Scrum and Kanban. Of particular interest is that when the interviewees described the related processes, the term "customer" was more often mentioned than the term "user" (Note that the absolute counts based on the verbatim transcripts are 247 and 54, respectively). This observation suggests that either the basic UCD principle of involving users has not been well realized or the two terms—customer and user—are employed interchangeably. It has thus motivated us to develop a survey to gather empirical data to shed more light on the compelling concern about a seeming negligence of users, or, in a broader sense, of UCD in general. The survey data also supplement the interview data on strengths and weaknesses of Scrum and Kanban.

The rest of the article is structured as follows: In section 2, we present our literature reviews on UX, Agile–Scrum, and Lean–Kanban. In section 3 we describe our conceptual analysis of the fitness of Scrum and Kanban to UX. The description is complemented by our report on empirical findings of the semistructured interviews with 10 practitioners in section 4. As stimulated by the results of the interviews, we conducted a survey of which the design, implementation, and results are delineated in section 5. The empirical data enabled us to identify a set of factors influencing the implementation of Scrum and

[2]In exploring the experiential component of Customer Experience, Gentile, Spiller and Noci (2007) derived the definition from the three resources cited: "The Customer Experience originates from a set of interactions between a customer and a product, a company, or part of its organization, which provoke a reaction. This experience is strictly personal and implies the customer's involvement at different levels (rational, emotional, sensorial physical and spiritual). Its evaluation depends on the comparison between a customer's expectations and the stimuli coming from the interaction with the company and its offering in correspondence of the different moments of contact or touch-points (LaSalle & Britton, 2003; Shaw & Ivens, 2005; Smith, 1999)" (p. 397).

Kanban in real-life contexts; we discuss these factors in section 6. In section 7, we reflect on the limitations of our research work presented in this article and their implications for future work. Finally, we conclude this article in section 8.

2. RELATED WORK

In this section we present concise literature reviews of the related work on UX and UCD, Agile and Scrum, as well as Lean management and Kanban.

2.1. User Experience

UX, broadly speaking, descends from the traditional UCD, focusing on the experiential aspect of HCI. In the field of HCI, there has been a shift of emphasis along several dimensions since about 15 years ago: from cognition to emotion, from pragmatic to hedonic, from productivity to experiential quality, from quantitative to qualitative methods, and some other evolvements (e.g., Bargas-Avilas & Hornbæk, 2011; Hassenzahl, 2004; Law & van Schaik, 2010; McCarthy & Wright, 2004; Vermeeren et al., 2010). In the meantime, the "dated" notion of usability has been replaced by the then-emergent UX, causing some confusion in the scope of research and practice, including job titles.

UX is still plagued with definitional and measurement issues (Law, Roto, Hassenzahl, Vermeeren, & Kort, 2009; Law, van Schaik, & Roto, 2014). Despite its crudeness, the standards definition, ISO 9241-210: 2010, which comes with three notes, is often referenced: *A person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service.* Essentially, the basic UCD principles are relevant to UX design, including the following:

- *Contextualization of users' goals, needs and tasks*: It entails observing users *in situ* and interviewing them to understand how a new system can improve the current situation and which specific tasks users can achieve with the new system.
- *User involvement*: Representative users should be involved in a range of activities all the way through the entire system development life cycle. This includes participatory design, formative and summative user-based evaluations.
- *Holistic design*: An overarching vision of core features of the system under development, especially the interdependencies among its components and the relations between such components and other contextual factors, can orientate the development team to a right direction.
- *User experience in totality:* The design of a system not only addresses the psychological aspect of the user but also takes into account the organizational, social, and economic factors that shape the user's experience with the system.
- *Iterative prototyping*: It is rare, if not impossible, to deliver a perfect design in the first release. Basically all design ideas are just assumptions that need to be validated with target users. With ongoing feedback from stakeholders, prototypes can be redesigned iteratively until they meet essential user requirements.

All these principles are geared toward the production of a usable, desirable system with useful and necessary features.

Some of the UCD principles, such as holistic design, are not compatible with Agile. Furthermore, the extent of user involvement (or user-centered activities) and the thoroughness of user-based tests (e.g., number of participant, test duration, number of iteration) are normally higher in UCD than in Agile (cf. the term "guerrilla style UX validation" suggests that discount methods are run in an impromptu manner). Common reasons for shortcutting user-based evaluation are tight deadlines (e.g., 2-week sprints) and staff shortage (e.g., UX specialists from a centralized UX department have to work for several teams simultaneously). Furthermore, with its emphasis on quick release and divide-and-conquer technique, Agile is deemed inappropriate to address some specific characteristics of UX, including the following:

- *Relevance of aesthetic experience*: A number of research studies have been conducted to investigate the relation between usability and beauty (e.g., Hassenzahl, 2004; Lavie & Tractinsky, 2004) and the role of aesthetic quality in UX (e.g., Hartmann, Sutcliffe, & de Angeli, 2008). Results thereof mostly confirm the relevance of aesthetic quality for user attitudes toward a product/system. Of particular interest is that one of the advocated changes for practicing Lean UX is "Speed first, aesthetics second" (Gothelf, 2013, p.116), based on the rationale that all design artifacts are primarily for communications and therefore no time should be wasted on beautifying them. However, it is empirically found that user expectation about the quality of an interactive system can easily be manipulated by (or is sensitive to) the appearance of the system, even in the early prototype phase. This illustrates the contradicting assumptions between UX and Lean.
- *Heavy reliance on data-rich qualitative methods*: As shown by a recent review on the UX work (Bargas-Avila & Hornbæk, 2011), qualitative methods are predominant, including narratives (cf. the dialogical sense-making approach; McCarthy & Wright, 2004), ethnographic observations, and interviews in order to develop a thorough understanding of users' actions, emotional responses, and associated motivations. Clearly, these methods are time-consuming and resourceful. Although in Agile and Lean "user stories"

are employed as substrate upon which the development work is built, they are normally not elaborate and stripped down to a bare minimum (i.e., one liner).
- *Temporality of UX*: As given in the ISO definition as well as in the UX literature, the trajectory of user experience can already start prior to any actual interaction with the object of interest. The six phases of sense making—anticipating, connecting, interpreting, reflecting, appropriating, recounting—underscore the need to work with users over time to make sense of their perceptions, cognitions, actions and emotions (Wright & McCarthy, 2010). Indeed, longitudinal studies (or at least medium term of about three weeks) are recommended for studying UX (e.g., Karapanos, 2013). This recommendation may clash squarely with the fast pace that the Agile and Lean approaches uphold.

Overall, integrating UX into Agile processes is deemed more challenging than integrating usability (cf. Lárusdóttir et al., 2009; Wale-Kolade, 2015) into the same processes, given a much broader scope of the former than that of the latter. Subsequently, we take a closer look of Agile in the form of Scrum and Lean in the form of Kanban.

2.2. Agile and Scrum

A major milestone in software engineering was marked in 2001 when 17 protagonists of lightweight software development processes such as Extreme Programming and Dynamic Systems Development Method gathered to share and discuss their visions. The seminal *Agile Manifesto* for Agile software development was then born (Gosper & Binnie, 2011). The Manifesto consists of four values and 12 principles that promote development, teamwork, collaboration, and process adaptability throughout the life cycle of the project (Beck et al., 2001). One of the core values in Agile development is *responding to change.* Given this value, developers accept that requirements can change and their challenge is to respond to the changes when these happen and not try to specify all requirements before coding. Another value in the Manifesto is *customer collaboration over contract negotiation*. This value is based on the understanding that it is not necessary or useful to specify contracted tasks in detail because requirements probably change throughout the system development process of a project.

Within the span of 13 years, a range of Agile approaches have been proposed and used. The Agile process Scrum has gained popularity in the software industry in recent years (VersionOne, 2014). According to its inventors, Scrum is the fastest and easiest to implement Agile process (Schwaber, 2009; Schwaber & Beedle, 2002). It provides a mechanism to improve the communication between developers and customers. In Scrum, self-organizing teams with mixed specialisms are emphasized, typically with six to eight interdisciplinary team members (Schwaber, 1995). In Scrum, the projects are split up in 1- to 4-week long iterations called sprints. At the end of each sprint, one (or more) *potentially shippable product* is delivered to customers. A characteristic of Scrum is the observation that small cross-functional teams can work effectively to produce good results.

Scrum typically consists of three major roles: a *scrum master* that acts as project manager/buffer to the outside world, a *product owner* that represents stakeholders, and a *team of developers* (less than 10). Some of the more important artefacts and ceremonies within Scrum are the "*product backlog*" of requirements to be managed by a product owner, the sprint backlog containing the requirements to be delivered after a particular sprint, and the daily stand-up meetings. The status of a sprint is often shown on a *burndown chart*, where all the tasks to be done are depicted in the beginning of the sprint. When some of the tasks are accomplished or "burned" over days, the number of tasks visibly left on the chart is reduced.

Scrum has been regarded as a process emphasizing UX, for example, by introducing user involvement through *user stories*, and by its iterative and communicative nature (Schwaber, 1997). This aspect of Scrum would then concur with the values of many UCD approaches, and this may then explain why Scrum has been successful. However, the UX work is not a mandatory part of the process and not something that can be taken for granted while applying Scrum in software projects (Da Silva et al., 2012; Salah et al., 2014). Even though many development organizations have reported great success of using Agile development processes, none of these processes explicitly specify that UX activities should be included in the process (Sohaib & Khan, 2010), and Scrum has particularly been criticized for not sufficiently integrating UX into the process (Singh, 2008).

One of the challenges mentioned by IT professionals who have considered UX in software development is that it has been hard to find time for UX activities such as user-centered evaluation (Lárusdóttir, Cajander, & Gulliksen, 2014). It has also been challenging for the IT professionals to maintain an overview of the total UX of the product in Scrum projects. It has been suggested that sharing documents, artifacts, and particularly knowledge between the development team and the UX specialists is one way of maintaining the overview or the big picture of UX (Beyer, 2010). Others suggested having a UX-knowledgeable person in a development team and more face-to-face communications between a UX specialist and other members in a cross-functional team (e.g., Kuusinen et al., 2012).

2.3. Lean and Kanban

Lean management originated from Toyota, where its principles were used to improve the efficiency in the production of cars, and the highest goal is to pursue "the absolute elimination of waste" (Ohno, 1988). Lean management is a holistic approach aimed at providing the right product in the correct amount of time at the right place with the right quality; it is

becoming more popular in both the public and private sectors worldwide (Modig & Åhlström, 2012). Efficient and effective production can be achieved by removing nonvalue adding activities in the production phase with the timeline of starting with receiving an order until ending with collecting cash. Shingo (1982) conducted a study on the Lean manufacturing and identified seven categories of waste. Poppendieck and Poppendieck (2007) mapped these categories to software development equivalences, calling it Lean software development.

The most popular process based on Lean management that is used for software development is Kanban, which has been used since 2004 (Kniberg & Skarin, 2010). Kanban has gradually gained popularity in software development (Diebold & Dahlem, 2014; Ikonen, Pirinen, Fagerholm, Kettunen, & Abrahamsson, 2011). The three fundamental principles of Kanban are (a) Limit Work In Progress (WIP; i.e., a new task should only be started when an existing one is delivered), (b) Visualize the workflow, and (c) Measure the lead time (i.e., the time it takes to finish one item).

The use of Kanban boards enables team members to see the progress of the project work and to have an overview of the tasks to be done. A Kanban board has at least three columns—*To Do, In progress*, and *Done*—as shown in Figure 2.

The project is then divided into suitable units and the names of the units are written on Post-It notes and put in the To Do column. Sometimes the In Progress column is divided into more columns like analysis, development, and testing. The number of units in the In Progress columns is limited, so the team is not allowed to work on more than a limited number of units each time. The process also aims at minimizing the average lead time, from the moment the work is started until it is done (Kniberg & Skarin, 2010). The Lean and Agile values of software developers were studied by Fagerholm and Pagels (2014) with a survey. Their analysis showed that Lean and Agile values are connected

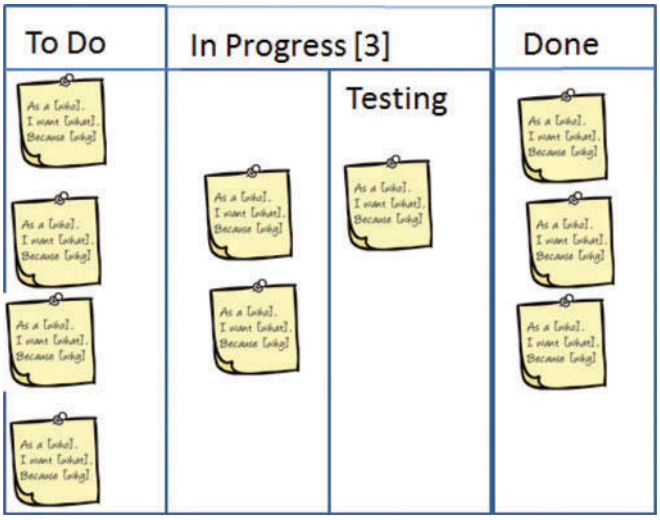

FIG. 2. An illustration of a simple Kanban board.

(e.g., broad stakeholder involvement, collaboration, openness to change, and flexibility), but not equal, to universal values (e.g., power, achievement, sense of purpose).

Overall, there have been some studies in the industry and the academic world to integrate UCD into Agile (e.g., Beyer, 2010), UX into Agile (e.g., Ferreira, Sharp, & Robinson, 2010), and UX into Lean (e.g., Gothelf, 2013). These efforts have led to some intriguing insights into the opportunities and constraints for that integration. In fact, the hitherto studies on Agile UX and Lean UX seem predominantly initiated by SE professionals. Rarely in these studies are the definitions, methodologies, and theories of UX discussed or even mentioned. This raises a compelling concern whether the UX community on the HCI side and the Agile community on the SE side should have engaged more in dialogues and idea exchanges.

3. ANALYTIC APPROACH TO ANALYZING SCRUM VERSUS KANBAN

As shown in the aforementioned descriptions, as Agile and Lean have different principles and practices, and Scrum and Kanban have specific characteristics, their respective compatibility with UX thus differs. To study the differences, we have adapted the comparison between Scrum and Kanban by Kniberg and Skarin (2010), who identified 13 characteristics instantiated differently in the two processes. However, they did not address any issue related to UX, nor did they organize the characteristics based on their shared concerns. To improve on their work, we categorized the characteristics into three groups—*Control, Team, Tool*—and analyzed them in terms of their fitness for supporting UX work (Table 1). Justifications for their relative fitness are discussed subsequently.

In the following we present our analysis of which of the two processes fits better for UX activities with regard to three aspects.

Control refers to rules and procedures applicable to the Scrum and Kanban approaches. Five characteristics are grouped in this aspect. (a) *Time-boxed iteration:* Sprints (ranging from 1 week to 4 weeks) in Scrum constrain the duration of UX design and evaluation activities. In Kanban the flexibility of the duration of each item (or "story point") can accommodate the UX work better; the Kanban team basically takes the time they need to complete an item. (b) *Item size*: In Scrum, breaking the work down into items with respect to time constraint can disrupt the holistic view and lose track of item dependency. In Lean there is no such constraint. (c) *Limitation of WIP*: The WIP limit undesirably leads to a tunnel (too focused) vision of the system that hinders UX professionals from giving effective feedback on design issues. (d) *Addition of new items*: Some UX activities such as user testing result in some extra work for a Scrum team, which is sometimes defined as a new user story. In that case, this task needs to wait at least until next sprint. (e) *Prioritization of product backlog*: In Scrum, the product owner prioritizes the key requirements on the product backlog.

TABLE 1
Comparing Scrum With Kanban in Their Fitness to UX

Characteristics	Scrum	Kanban	Fits for UX
Control			
• Time-boxed iterations	Prescribed	Optional	Kanban
• Item size	Prescribed (feasible to be complete in one sprint)	Not prescribed	Kanban
• Limitation of work-in-progress	Indirectly (per sprint)	Directly (per workflow state)	Neither
• Addition of new items	Not allowed for an ongoing iteration	Allowed if capacity is available	Kanban
• Prioritization of product backlog	Prescribed	Optional	Scrum
Team			
• Team commitment to a specific amount of work for an iteration	Prescribed	Optional	Kanban
• Team composition	Cross-functional team prescribed	Specialist teams allowed	Both
• Ownership of the visualized information of work progress	A sprint backlog owned by one specific team	A Kanban board shared by multiple teams	Kanban
• Role assignment	Prescribed roles (product owner, scrum master)	No role prescribed	Neither
Tool			
• Default metric as monitoring tool	Velocity	Lead time	Neither
• Graphical support tool	Burndown chart prescribed	None	Neither
• Estimation of progress	Prescribed	Optional	Kanban
• Persistence of the information on work progress	A Scrum board is reset between each sprint	A Kanban board is persistent	Kanban

Note. UX = user experience. Adapted from Kniberg and Skarin (2010, p.50).

If the prioritization takes the view of the UX professional into account, it can facilitate the work of UX.

Team refers to the nature and tasks of typical Scrum and Kanban teams. Four characteristics are grouped in this aspect. (a) *Team commitment to a specific amount of work for an iteration:* Although the prescribed scope of work enables the predictability of the project's process, it limits the malleability of reallocating resources. Negative impact, if any, can become acute if the extra work (i.e., a series of items) to be addressed is interdependent with those within the current iteration. Typical trade-offs between prescriptiveness and adaptiveness are relevant. (b) *Team composition*: Both strategies can benefit the work of UX professionals, who typically have background training in psychology, given their main role in coordinating user studies and providing consultancy for user-related issues. A cross-functional team allows a UX professional to be colocated with developers and other team members, enabling close collaboration on a day-to-day basis. In contrast, UX professionals in a specialist team can provide centralized services (e.g., user research, interaction design) to different development teams company-wide, thereby providing coherent UX initiatives and building up a strong identity of their role. (c) *Ownership of the visualized information of work progress*: A sprint backlog focuses only on the functionality described on it. A shared view of the ongoing work with a Kanban board can provide UX professionals a more holistic perspective. (d) *Role assignment*: UX professionals may not fit very well into the three specified roles of Scrum—product owner, Scrum master, and developer. Presumably, a product owner with training in UX could facilitate the integration of UX in Scrum. In Kanban the roles are not prescribed, UX professionals may fit better in the process. Hence, neither Scrum nor Kanban seem to have a clear advantage for UX on this characteristic, depending on the profile of individual team member.

Tool refers to the tools for supporting the Scrum and Kanban processes. Four characteristics are grouped in this aspect. (a) *Default metric as monitoring tool*: The ambition to attain a high Velocity (as the performance measure of a Scrum team) may drive the team to lower the priority of user-based studies critical for UX design. Similarly, shortening Lead Time may compromise the effort that a Kanban team may ascribe to user-based work. (b) *Graphical support tool*: Burndown chart is too development oriented. A diagrammatic tool illustrating the changes of user acceptance over time would be more relevant to UX people. (c) *Estimation of progress*: In Scrum the work on user stories is estimated in terms of story points. Similar to the issue related to Default Metric, the time pressure imposed may compromise resources for the UX work. (d) *Persistence of*

the information on work progress: The Scrum team loses references about previous sprints, and UX professionals cannot track the interplay between user evaluation and system redesign. As the Kanban board is not reset, such references remain accessible.

To summarize, our analysis indicates that in general Kanban fits better for UX work than Scrum does. In particular, the characteristics pertaining to Team favor Kanban. Apparently, the flexibility of Kanban gives its advantages over its constraint-laden (or overstructured) counterpart, Scrum, to enable the integration of UX into software development processes.

4. STUDY 1: INTERVIEWS WITH PRACTITIONERS

Based on the literature review and on the conceptual analysis of the characteristics of Scrum and Kanban, we argue that UX activities are better integrated into Kanban than Scrum. Nevertheless, we aimed to explore further whether our conceptual understanding of Scrum and Kanban matches the practices in reality. To meet this aim, we performed a *secondary data analysis* of semistructured interviews with 10 practitioners in software industry in Iceland where the uptake of Scrum by IT professionals has been relatively high (Lárusdóttir et al., 2009). Secondary data analysis is the analysis of data that have been collected by people other than the researchers analyzing the data or that have been collected for purposes other than the one under consideration, or it is a combination of both (Vartanian, 2010). In the current study, the second author was involved in the data collection with the interviews, which were aimed at discussing the software development life cycles deployed in different industries. As Scrum and Kanban have been adopted in the companies where the interviewees worked, the interview data became coincidentally relevant to the current study.

4.1. Methods

4.1. Participants and Procedure

All the 10 semistructured interviews followed the same structure. First, the interviewees (P1 . . . P10; two female, eight male) were asked to describe some basic information about their organization and their role in it (Table 2). The business areas of the organization are rather diverse, including enterprise content management, government applications, banking, Agile consultancy, telecommunication, resources management, web services for healthcare, market data management, and airline. The interviewees were also asked more specific questions of their experiences concerning the software development process adopted in their company, especially how they involved customers and end-users. Five of our interviewees used both Scrum and Kanban in parallel, sometimes one for each project that they worked on simultaneously. Two interviewees adopted Scrum only, and three adopted Kanban only.

The interviews were conducted mostly in the interviewees' organizations. Two of the interviews were conducted at a university. All interviews were conducted in English and audiotaped. On average each interview lasted about 45 min, during which the interviewee (P) was asked a set of questions (Table 3; Q.1.1-Q.1.5: background questions; Q2.1- Q2.7: specific questions on managing the software development process).

All recorded interviews were transcribed verbatim. The data from the interviews were compiled, analyzed, and coded. Content analysis techniques (Krippendorff, 2004) were applied with both the top-down approach with predefined themes derived from the related literature and the bottom-up approach for generating subthemes. This was an iterative approach, and the subthemes were refined with every transcript of the interviews.

4.2. Results and Discussion

Results are presented in three subsections: the application of the processes: Scrum, Kanban, or both; strengths and weakness of the processes; and collaboration with stakeholders during the processes.

Applications of the Processes

Interviewees applying Scrum only. The application of Scrum in the two cases (Table 2) was a sort of a "standard Scrum practice," as described by P3. Both interviewees (P2, P3) ran sprints of 1 to 3 weeks, with about seven people on the teams. They used the sprint backlog, and the team members picked the tasks to work on from there. They pointed out that the retrospective meetings were useful for improving the process of their work and that daily stand-up meetings served as a good forum for knowledge sharing. P3 explicitly remarked that every single bit of code went through peer review.

Interviewees applying Kanban only. The three interviewees described that they used Kanban boards (Figure 2) to visualize the on-going work of the software developers. The interviewees mentioned that the number of current tasks for each developer was typically restricted from one to three. The visualization of what the other members were doing and the responsibility for the tasks encouraged communication within the teams. P6 described that they held stand-up meetings (which was actually a Scrum activity) to discuss the information on the Kanban board; this enhanced communication in the developer group. P4 mentioned that they used Kanban because most of the projects lasted 100 to 200 hours, and Scrum would be too heavy for projects of this size.

Interviewees applying both Kanban and Scrum. P8 used Scrum in "bigger" projects and Kanban board for visualizing the status of these projects. They also used the Lean principle of restricting the number of tasks each developer was working on to two. This was similar to what P10 was doing. They broke the project down in the minimal marketable features (MMFs), which could be finished one by one. Scrum was used for the process of developing and executing each MMF, and Kanban was used for getting an overview of the MMF and to prioritize which MMF the teams were working on at a particular

TABLE 2
Overview of the Profiles of the Organizations and Roles of the Interviewees Therein

Process Used	Code	Interviewee's Company Profile	Role of Interview in the Company	Years in Software Development
Scrum	P2	∼50 employees; 17 years old, Serving external customers	Managing director;	20+
	P3	∼ 90 employees; 28 years old; Serving external customers	Product team lead;	20+
Kanban	P4	Over 1,100 employees; > 60 years old; Serving external customers	Director of software development;	10
	P5	25 employees; 19 years old; Serving external customers	Project manager;	20
	P6	4 employees; 5 years old; Serving external customers	Product owner and programmer	20+
Scrum and Kanban	P1	37 employees; 5 years old; Serving external customers	Director and product owner	12
	P7	Over 1,100 employees; >60 years old; Serving external customers	Agile consultant	12
	P8	> 1,000 employees; >100 years old; Serving in-house customers	Department manager	12
	P9	12 employees; 7 years; Serving external customers	Project manager	20+
	P10	> 600 employees; ∼100 years old; Serving external customers	Scrum master	20+

TABLE 3
Questions of the Semistructured Interviews With 10 Practitioners in the Software Industry

No.	Question
1.1	Describe the field of operation of your company.
1.2	What kind of software development process do you use? When did you start with this approach?
1.3	What is your experience with other approaches?
1.4	What is the structure of the teams working in software development?
1.5	How do you define the term "waste" in your own words?
2.1	How do you describe "partially done work" in your organization?
2.2	Describe the process of requirements engineering. Do you have the impression that you would implement more feature than demanded? How are requirements managed?
2.3	Do you think knowledge gets lost in the progress of a project, which makes relearning necessary?
2.4	Describe handoffs in your development process.
2.5	How are tasks assigned?
2.6	How many tasks has a developer to process at a time on average?
2.7	What are the main reasons for delays in the development process?

time. The team used sprints, sprint backlogs, and user stories for each MMF. P1 described that they used Scrum for project management and Kanban for managing the maintenance process, so they actually had two processes running, one for each type of project. This interviewee mentioned that at the end of the retrospective meeting (a Scrum activity) the team decided one issue to improve during the next sprint, which was a Kanban way of thinking about prioritization. P7 told a similar story that they used Scrum for software development projects and Kanban for improvement projects on a Kanban wall. P9 used more a mixture of Scrum and Kanban. In his company they worked in teams and described user stories (Scrum principles).

The user stories were managed by a Kanban board, and they also constructed a story map for bigger visions at a 6-week pace. They committed to continuous delivery of completed features and worked on one feature at a time (Kanban principles). They also emphasized continuous improvements by allocating one afternoon each week to their work on improvements for the company. P8 mentioned that in Scrum, there was a lack of clear vision for the project, so it was hard to maintain the holistic view of the project, as in Scrum the project was divided into many small chunks in the sprints. This interviewee also mentioned that it was hard to react on changed requirements in Scrum, as the team had started to work on new things in the new sprint, when the changes in the requirements were communicated and the customer had to wait for at least one sprint for the developers to start working on the changed requirements.

Impacts of Scrum and Kanban. From the transcripts we identified the strengths and weaknesses of Scrum and Kanban practices and categorized them into four types of impact—*organizational, economic, cognitive, and social*. These aspects have commonly been identified in analyzing different system development processes within organizations (e.g., Hofman, 2011; Olson & Olson, 2003; Van Dyne & Pierce, 2004). In addition, some interviewees discussed possible reasons and constraints for the success or failure of realizing Scrum or Kanban. Furthermore, we have coded the related responses with emerging subthemes using self-explanatory labels. Each of the subthemes listed subsequently was mentioned by at least three interviewees. To illustrate certain points vividly, wherever appropriate, we cite verbatim the utterances of the interviewees (P1 . . . P10), who were non-native English speakers. An interesting observation is that the interviewees described similar opinions and experiences about Scrum and Kanban, irrespective of the size of the company where they worked, their role in the company, or their software development experience (Table 2).

Scrum: Organizational impacts. Three strengths and four weaknesses of Scrum from the organizational perspective have been identified. Paradoxically, the same characteristics can be regarded as strengths and weaknesses at the same time. As Scrum is essentially a process-oriented tool, its impacts on organizational issues are relatively more salient.

- *Visibility (or transparency)*: "the best thing both Scrum and Kanban is the visibility of what others are doing . . . you know right way to see if something's wrong." (P4)
- *Synchronization of different processes*: "We have a separate team, for example with Agile coaches that try to streamline or synchronize how teams in different products are working, so we should have a similar view. . . . So one team is not working totally different than we are, even though we have a lot of tweaks how we structure." (P2)
- *Continuous improvement through retrospectives*: "Internally we try, and this is also part of Scrum, we do retrospectives and we try to feed that back into our backlogs of task lists. So that is one thing we try to improve individually each team. We also try to cross team, we try to have and share good or bad points regularly as well; we also try to do post mortems. . . . So yeah most important for the software process are the retrospectives, how we use that." (P2)
- *Impractical alignment between sprints and release cycle*: "Not linking release to sprints because the sprint and release cycle are different; the sprint team should be kept waiting and must move on to another sprint when something else to be done for the almost-ready release; Difficult to measure continuous improvement in terms of story points as they are always 'flow-overs' to the next sprint." (P3)
- *Limited by project size and duration*: "For a small project with a short lifetime it is impractical to have a five-person cross-functional team. We have no money for that." (P1)
- *Unrealistic expectation about fixed team composition*: "It is sometimes impractical to keep everybody in a team together all the time, because members with special expertise and experience are needed elsewhere." (P1)
- *Inflexibility of sprints*: This constraint leads to several issues: (i) changes in requirements cannot be accommodated; (ii) inappropriate for bug-fixing that needs immediate attention; (iii) long waiting time for bug-fixing because of the big backlog; (iv) idle time in the development team when waiting for the dependent tasks to be completed.

Scrum: Economic impacts. Scrum is perceived to have only positive impacts on the economic aspect. In fact, one main rationale underlying the creation of Agile processes was to improve the value—financial as well as personal—of software development.

- *Value-driven continuous delivery*: "Every six weeks, we do the bigger picture. . . . It's all about prioritizing value to them [developers] . . . but for every week, we work on the Kanban board, and we have user stories. . . . An item of value to them that has to be done. . . . If it takes more than 3 days we try to break it up, but it's something that can be delivered, all the way. And it's value to them if it comes into operation. . . . We have a continuous delivery pipeline . . . . So within a span of a week, we maybe deploy to production 2 or 3 features, pieces of value. So that's how we work with value. (P9)
- *Faster financial return with waste reduction*: " . . . we had too many projects ongoing, so it was problem that we didn't finish them. So with breaking them down into smaller projects, MMFs, then we can finish them easily . . . , maybe the whole project here is not finished, but this MMF, that is a value on the market, that

we can get some money back, then we are happy with that outcome. Because otherwise it would be a waste, it would by lying here and not deployed and we started working on the other one." (P10)

Scrum: Cognitive impacts. A particular characteristic of Scrum—daily stand-up meetings—is deemed as a powerful tool for stimulating knowledge exchanges and knowledge building. Scrum is regarded to have only positive cognitive impacts on software development teams.

- *Enabling reflection, knowledge sharing and transfer through retrospectives and stand-ups*: "Every single bit of code goes through peer review. . . . A huge knowledge sharing process." (P3); "When one piece of work goes from one person to another, there is also some kind of knowledge transfer involved. We have the stand-up meetings and we are in good communication with everyone, so everyone knows what everyone is doing, so these transfers are rather simple." (P6)
- *Accelerating the process of specialization*: "We talk about it during stand-ups. . . . We say, 'I did something similar last week, I can take this, I'll be quicker than you.' or maybe somebody asks, 'Would you like to take that?' But it's definitely not assignment. It's collaboration . . . . Specialists would evolve out of this. . . . They will be very specialized in a certain part of the software." (P3)

Scrum: Social impacts. A visible type of impact of Scrum is its role in shaping social relationships among the members of a development team, who are obliged to communicate and collaborate on a regular basis. Three positive and one negative of such impact have been identified.

- *Enhanced team spirit*: "We have daily meetings at 10 to 12 before lunch. We go through where everyone's at, so everybody knows what everybody is doing. That's great. Because it gives you more sense of a team. It gives me happier employees." (P4)
- *Stronger developer commitment*: The pull strategy enables developers to choose their own tasks from the backlog of prioritized user stories, inducing commitment in developers.
- *More direct and effective communications enabled by daily stand-ups*: " . . . We usually try to schedule just half an hour meeting, because then programmers talk together. It's usually no problem, but if they are talking through email or Jira they tend to ignore each other and make stupid comments. But when people meet there are usually no problems and that's resolved much trouble." (P10)
- *No direct contact between developers and customers*: It is because communication between the parties is mainly mediated through the product owner serving as a gatekeeper of information. In this sense, developers are "protected" from being interrupted by customers' ongoing demands.

Kanban: Organizational impacts. Three positive (the first three points) and three negative impacts of Kanban have been identified.

- *Visibility and traceability*: "Someone was supposed to do something, but hasn't done it. So more like that. These are really obvious by using Kanban, which is a really good thing." (P4)
- *Appropriate for open tasks*: "We sometimes we switch to Kanban, if we are doing maintenance or during regression testing where we have unknown work coming up." (P2)
- *Jam in backlog*: "We also have a very classical lean problem. We have too many features in the inventory that are not in the testing environment and are in there for too long. So we are focusing on reducing this. Unfinished features, partially done work." (P1)
- *Matching right people with right tasks*: "That's also Scrum and Kanban, but there are loads of things we want this person to do, not that person, things like that. So that's missing." (P4)
- *Optimizing the number of tasks per developers*: The related challenges are to limit the number of projects that are being worked in parallel and to avoid developers getting too many tasks at one time.
- *Lack of testing protocol, especially for the aesthetic aspect*: "We would all just go to the staging server and test all the features that had been gone through the iteration. . . . We are now thinking about adding the new category that says, it's been tested move it to the system. It's been manually tested, doesn't mean visually look good. Our testing process doesn't test aesthetics. . . . There is no protocol that we are going through. I'm doing it twice, I think we can do it just once, so we add another category." (P6)

Kanban: Economic impacts. Similar to Scrum, only positive economic impacts of Kanban have been mentioned by the interviewees.

- *Shorter throughput rate of features and fixes with continuous integration, deployment, and delivery*: This can be enhanced by taking fewer features and focusing on finishing them. A high granularity of items or user stories (e.g., 2 days to work on each)—elements of value to deliver; "The aim of lean is to enable the flow of projects as fast as possible." (P5)
- *Fast feedback from customers*: "Just to show and have feedback, and have the customers to try it out and so you deploy a little, so you get feedback to your development. That is the most important rather than to wait and build stuff that nobody uses, which happens a lot." (P9)

Kanban: Cognitive impacts. Two positive and two negative impacts have been identified. This pattern of perception is different from that of Scrum.

- *Mitigating knowledge loss and facilitating knowledge flow through daily reporting and follow-up discussions.*
- *Raising the awareness of the quality level in the team.*
- *Noninterchangeability of specialists*: "When you are doing Kanban that all programmers are interchangeable, which they aren't always. . . . They can be enormous differences." (P4)
- *Undesirable context-switching and interruption for specialists*: "So context switching is a problem, especially for certain team members that are very knowledgeable. They will always be interrupted, because they are the guys that can answer the questions or help you fix things. Definitely a problem for those people, because they will simply be the ones people go to. . . . People usually tend to work on two things. So you have maybe one thing, if you are waiting for something, then you switch to something else to fill the time. . . . Such context switching leads to lower productivity." (P2)

Kanban: Social impacts. Four positive impacts and one negative one have been identified. This pattern is similar to that of Scrum.

- *Encouraging communications through daily meetings.*
- *Promoting collaborative efforts for unblocking problems and prioritizing work.*
- *Enhancing a sense of task ownership in developers*: "But with the Kanban, we have had this problem of developers getting too many tasks at a time. But we try to limit it with the Kanban wall, we have a system where we had this magnets, with the faces of developers. And they only get to have three projects open at a time . . . so if we see someone has all his faces out and tries to get in another project we try to get in and have it resolved. . . . It also has a certain psychological aspect of it, because they see their faces and 'I am working on this project' so that means something and everybody can see it." (P5)
- *Help reducing the frustration caused by the changes of priorities*: The flow of MMFs is regulated by Kanban and the senior management can't change the priority on the fly once a MMF is started.
- *Repetitiveness and boredom*: "Kanban is getting a bit, everybody is coming to the meeting and it is the same old stories . . . . try to change a little bit . . . to keep things dynamic and interesting, not boring." (P4)

Stakeholder Collaborations

P2, who used Scrum only, explained that the customers were fairly isolated from the development and that the customers complained that they did not get a quick service. If the requirements were changing frequently, the customers did not want to wait for 2 weeks until the sprint was over and the team started to work on a new sprint, including the changing requirements.

P5 and P6, who used Kanban only, remarked that it was hard to control the communication between the customer and the developers. P6 explained that developers felt like getting "noise" from the customers and the product owner, and consequently they made features that had not been thought through. This had the negative effect that the developers started responding to requests from the customers, without looking at the holistic effect of the change. P5 mentioned that there was no obvious value for the customer because the increments were very small when using Kanban.

P9, who used both Scrum and Kanban, pointed out that fast feedback was expected from customers:

> . . . just to show and have feedback, and have the customers to try it out (the new feature) and then you deploy a little, so you get feedback to your development. That is the most important rather than to wait and build stuff that nobody uses, which happens a lot.

P1, who also used both Scrum and Kanban, explained that more direct communication between the customer and the developers was needed. The rationale was that the business analysts talked to the customers and the business analysts made user stories and some information was lost in this chain. P8, who also used both processes, told a similar story. He thought that there was a lack of communication with users when using Scrum and that the users were not involved early enough in the development process. This interviewee explained that the product owners were supposed to understand the user's requirements, but sometimes they did not and some useless functionality was developed. P7 described that from the customer's point of view the service was decreasing because in Scrum the goal was to protect the developers. This interviewee suggested that customers should be trained to be more actively involved in giving feedback to developers.

An intriguing observation is that some of our interviewees talked about customers and users as if they were synonyms. When asked about how bugs were typically handled in a company, our interviewees responded, "So we can handle it with just one employee in user support using email. We do not get many complaints from customers." This interviewee used both user and customer interchangeably when explaining how they handled bugs. Another interviewee explained the importance of involving the customers by saying,

> And especially just to show and have feedback, and have the customers to try it out and so you deploy a little, so you get feedback to your development. That is the most important rather than to wait and build stuff that nobody uses, which happens a lot. Or maybe only 30% of what you did will get used.

This interviewee also referred to customers and users of the system as if they were synonyms. Another example is that an interviewee explained how important it was to talk to users:

> So talking to the user, people tell us all the time and we try to do that a lot, like we are having them sitting with us and also we tend to try to have meetings with them and try to get them in as often of possible. Trying to be on the track that the customers find it agreeable, it's not enough to be on our own track the customer has to be there with us.

This interviewee also used the terms *users* and *customers* as if they were synonyms.

Nonetheless, in practice development teams tend to minimize communication overhead by working with one customer instead of with a group of users. This distancing strategy aims to reduce the risk of constantly changing requirements. A customer is someone who owns the product, whereas a user is someone who is affected by it when using it (cf. Table 4). In Agile, assuming a customer to play these two roles is incompatible with the UCD philosophy (Section 1). When evaluating the impact of a product on users becomes the *sole* responsibility of a customer, the validity of evaluation results will be compromised. If the customer is not in touch with the user base, the evaluation will primarily reflect his or her personal judgment. Even if the customer collects feedback from some end-users and reports back to a development team, the filtering effect caused by such a customer-mediated process may lead to the loss of some important user-based feedback.

5. STUDY 2: SURVEY WITH PRACTITIONERS

As described in Section 4, the usage of the terms *customer* and *user* is confusing and nondiscriminant. This is consistent with the comment about the peculiar semantics of Scrum (Beyer, 2010; Illmensee & Muff, 2009). Agile has indeed redefined the key word in the business world—customers—from ordinary people who buy things from retailers to professionals who manage the purchase of a product or a service. We were motivated to explore the issue of terminology confusion with practitioners from software industry. In addition, we aimed to gather more data on the practical experience of deploying Scrum and Kanban. To enable a wide coverage of potential respondents from a spectrum of sectors, we chose to implement a web-based survey.

5.1. Method and Procedure

The survey consisted of 48 questions, 40 close-ended and eight open-ended. The first part of the survey collected information regarding the participant background, work environment, and experience. The second part contained four questions regarding the participants' use of software development processes and their preferences. The third part contained a combination of 14 open-ended and close-ended questions to gather their understanding of the terms *user, customer*, and *client*. In addition, we asked them to describe the people belonging to

TABLE 4
Definitions of Users, Customers, and Stakeholders From Different Sources

User/End-User	
Person who interacts with the product.	ISO 9241-11:1998
The person who interacts directly with the product or system to produce a desired result.	Beyer (2010)
A person who uses or operates something.	Oxford Dictionaries
Customer	
Customer organization or person that receives a product or a service. Note 1: A customer can internal or external to the organization; Note 2: Other terms commonly used for customer are acquirer, buyer and purchaser acquirer - stakeholder that acquirers or procures a product or service from a supplier; Note 3: The acquirer could be one of the following: buyer, customer, owner, purchaser;	ISO/EC 12207:2008
People who derive value from the system: users' management, the purchases, possibly the IT department that has to maintain the system, and anyone else who has a say in adopting the system.	Beyer (2010)
A person who buys goods or services from a shop or business.	Oxford Dictionaries
Stakeholder	
Individual or organization having a right, share, claim or interest in a system or in its possession of characteristics that meet their needs and expectations.	ISO/IEC 15288:2008
People in the development organization who depend on the system being correct. Management of the development team, the product owner, and the marketing team are all potential stakeholders.	Beyer (2010)
A person with an interest or concern in something, especially a business.	Oxford Dictionaries

these three groups in their development project. Finally, the survey had questions about gathering feedback on design artifacts from users, customers, clients, colleagues, and friends.

5.2. Participants

The survey was web-based and distributed via e-mail to 393 graduates from Computer Science of Reykjavik University who all had successfully completed at least a B.Sc. program there and graduated between 2009 and 2014. Out of these graduates, 73 responded to some questions in the survey (i.e., the response rate of 18.6%). All of the participants received more or less the same training in computer science, and all of them had completed a course in HCI as a compulsory part of their education. Of the respondents, 81% had completed a B.Sc. degree, 12% had completed a M.Sc. degree, and 6% had a Ph.D. degree. One respondent did not answer this question. None of them participated in the interviews of Study 1.

Seventy-two participants responded to the first 16 questions of the survey (86% were male; 14% were female), and one only responded to the first eight questions of the survey. That respondent did not answer the question on the process usage and was therefore dismissed from further analysis in this article. Their industrial experience varied: 10% graduated in 2009, 14% in 2010, 21% in 2011, 17% in 2012, 18% in 2013, and 21% the same year of the survey.

The largest group of the participants (30%), at the time the survey was administered, worked in a company that had more than 200 employees. For the size of the company, 26%, 19%, 14%, and 11% of the participants worked in a company that had fewer than 21, 21–50, 51–100, and 101–200 employees, respectively.

To the question of how many people were involved in developing software in their workplace, 58% of the participants responded that there were fewer than 21 people, 14% mentioned 21–40 people, and 12% mentioned 41–80 people. About 16% of the participants said there were 81 or more people involved in software development in their company.

The participants were asked about their main job role and were allowed to choose more than one. About 83% of the participants selected programming as their main task in their workplace. Other options included design (53%, the second highest), requirement analysis (∼31%), and software testing (∼31%). Concerning the types of software developed in the participants' companies, the sector "Business/Finance" was the most common (31%), followed by "Data Management" (Figure 3).

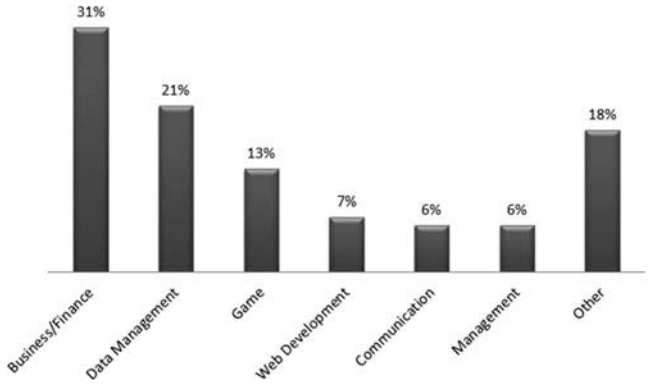

FIG. 3. Distribution of the main type of software that the respondents developed.

5.3. Analysis of the Survey Data

In the survey, respondents were asked two open-ended questions (Q.17 and Q.21) with similar wording: "*Please describe what the concept 'user' ['customer'] means*." In assessing whether the respondents interpreted these two concepts appropriately, we reference the definitions from three sources: ISO standards, Beyer (2010), and Oxford Dictionaries (online), which represent business-oriented, academic-oriented, and everyday usage (Table 4). For the sake of comparison, we also included the definition of a closely related term, *stakeholder*.

The two authors coded the respondents' definitions independently based on a simple scheme: "Correct," "Incorrect," and "Unclear." A response was coded as correct (incorrect) if it was semantically consistent (contradictory) with any of the definitions given in Table 4. If a response was too broad or incomprehensible, it was coded as unclear. The intercoder reliability (Cohen, 1968; Fleiss, Levin, & Paik, 1981) was high, with that for the term *user* (Cohen's $\kappa = 0.83$) being about equal to that for the term *customer* (Cohen's $\kappa = 0.79$).

5.4. Results of the Survey and Discussion

In this section, we first present and discuss the results of the terminology usage, which were grouped according to the system development processes used: Scrum only ($n = 25$), Kanban only ($n = 8$), Both Scrum and Kanban ($n = 14$), and Others ($n = 24$). Then we describe and analyze the findings about strengths and weaknesses of Scrum and Kanban.

Defining the Terms User *and* Customer

Although there were altogether 73 respondents, only 40 responded to these two questions. We speculated the reason for nonresponse was *not* due to its being an open-ended question (Reja, Manfreda, Hlebec, & Vehovar, 2003) but rather to the respondents' lack of interest or even relevant experience. The subsequent questions (Q.23-Q.46, close-ended questions) of the survey were related to the collaborations with users and customers, and the response rates to them dropped even further. Table 5 shows the distribution of response and nonresponse rate for the terms *user* and *customer* with respect to the software development processes the respondents predominantly used in their workplace.

TABLE 5
Distribution of the (Non-)Response Rate for the Definition of User and Customer

	User		Customer	
Process Used	Response	Nonresponse	Response	Nonresponse
Scrum only[a]	16	10	16	10
Kanban only[b]	4	4	4	4
Scrum + Kanban[c]	6	8	6	8
Others[d]	14	10	13	11
Total	40	32	39	33

[a]$n = 26$. [b]$n = 8$. [c]$n = 14$. [d]$n = 24$.

Defining the term user. Table 6 shows the results on the correctness of the responses to the question defining the term *user*. Sixteen respondents using only the Scrum process responded to this question. Ten of them explained it according to the definition from ISO or in similar terms. Four gave unclear definitions, including "End-user, system admins"; "Any type of end user of the software"; "Customer or user of the product"; "Users might also be a computer program that might be intended for users." The response "Someone I need to nurture and hope doesn't hurt him-/herself" was coded as incorrect because it is incompatible with the respondent's job of developing a business web application.

Six respondents who were using both Kanban and Scrum responded to this question. Three of them had the right definition, and three gave a too broad (unclear) definition: "End user," "Business," and "Those that use and work using the system. That is almost everybody are users in one way or another." Four who were using only Kanban responded to this question. One of them had a generic (unclear) definition: "Multiple people using some part of our system. Also system using another system can be considered a user." The other three defined the term according to the definition in ISO 9241 standard.

Out of the 14 using processes other than Kanban or Scrum who responded to this question, four gave an incorrect definition: "Nothing really, as we are trying to discover ways to solve certain problems regardless of how they eventually will be used in actual application," "An individual behind a computer that has low to medium computer skills," "A multilayer object that initiates a request," and "If you don't know this, you're studying the wrong field." Seven gave a definition similar to the ISO standard, and three gave a too broad definition: "The customer that uses the product," "Either a physical person or a system/computer/program that uses your program/system," and "I have seen users defined as pretty much anything from a single person to another organisation or a piece of software."

Defining the term customer. Table 7 shows the results on the correctness of the responses to the question defining the term *customer*. Fifteen respondents using only the Scrum process responded to this question. Ten of them explained the term "customer" according to the definition in ISO/EC 12207:2008. Many of them mentioned that it is someone who pays for the product they are developing or the software. One explained that it is "someone who buys something," which we interpreted as a too broad definition. Another one explained that a customer is "a company or a department in a company" without specifying the function of such a company or a department (cf. Table 4), and we interpreted as incorrect. Of interest, three respondents explained that customers are the same as users saying: "same a user, software for in house use only," "same as user," and "consumer or somebody using the product." In addition, in response to a close-ended question "Is a user the same as a customer?"

TABLE 6
Distribution of the Type of Response to the Definition of the Term *User*

	Correct	Incorrect	Unclear
Scrum only[a]	10	2	4
Kanban only[b]	3	1	0
Scrum + Kanban[c]	3	3	0
Others[d]	7	4	3
Total	23	10	7

[a]$n = 16$. [b]$n = 4$. [c]$n = 6$. [d]$n = 14$.

TABLE 7
Distribution of the Type of Response to the Definition of the Term *Customer*

	Correct	Incorrect	Unclear
Scrum only[a]	10	2	4
Kanban only[b]	3	1	0
Scrum + Kanban[c]	3	3	0
Others[d]	7	4	3
Total	23	10	7

[a]$n = 16$. [b]$n = 4$. [c]$n = 6$. [d]$n = 15$.

(Q.23), three of these 16 respondents chose "yes" and 12 chose "no."

Six respondents who were using both Kanban and Scrum responded to this question. Three of them had the right definition of the term *customer*, one wrote, "Don't know," and two had definitions similar to the definition of a user saying, "Depends on context. Customer of my company (aka client) or customer of the product (aka the user)"; "All the users of the product." When asked directly if a user is the same as a customer, three answered "yes" and three "no."

One of the four respondents who used only Kanban defined customer according to the definition in ISO/EC 12207: 2008. Two defined it as "a customer is some one that is a potential buyer" and "People who bug me." (The respondent might want to sound humorous or sarcastic.) And the third said, "User of our products." When asked directly if a user is the same as a customer all of them chose the answer "no."

Thirteen respondents used processes other than Kanban or Scrum. Eight had the right definitions, with the explanation that a customer is the person buying a product, and one explained it was "the owner of the product." Two answers were not applicable and two were too broad: "A person who buys goods or services from a shop or business" and "A person that does not know what they want and is generally never happy." (This respondent seemed to be expressing his frustration.) One explained that "in my opinion this is more related to commercial products, somehow overlaps with the user, but by saying customer it is implied that it can be a user that pays for using the application." When asked directly if a user is the same as a customer, two responded "yes" and 11 "no."

Overall, the respondents had a reasonable understanding of the distinction between user and customer, given their references to the definitions in the ISO standards. The data (Table 6 and Table 7) suggest that those experienced in Scrum seem to have better knowledge of "user versus customer." However, the numbers are too small for any inferential statistics. Nonetheless, it is still worrying that some respondents tend to equate users with customers, who are different target groups with different needs, values, expectations, and preferences. On the other hand, as 55% of the respondents have worked in industry for less than 3 years, they might have relatively limited experience in dealing with customers and users directly; this could influence their understanding of the two roles in practice.

Strengths and Weaknesses of Scrum and Kanban

In the survey, the respondents were asked to specify which software development process they had been using and which one they would prefer to use. Of the 25 respondents using Scrum, 22 wanted to continue to use the process. When describing the reasons for the continuous use, the respondents described the strengths of the process. The most frequently mentioned strength of Scrum was its effectiveness, because a finished feature would be delivered in each sprint. In addition, some mentioned that the short sprints of Scrum helped the team members to focus on a limited number of things at one time and that it was easy to plan and monitor individual as well as team progress. Others mentioned that the cooperation in the teams made everybody involved in the software development. Some preferred Scrum because of the continuous improvement process through the retrospective meetings. Some said that they preferred Scrum because it worked and they knew the process well. One respondent currently using Scrum wanted to shift to the Lean process, because there would be closer contacts with users and there would be no paperwork that would not add value.

Of the eight people who were using Kanban only, seven wanted to continue to use it and one wanted to switch to Scrum. The reasons for the respondents to use Kanban were mainly because of its simplicity, visibility, and low overhead. Some also mentioned that it fitted small teams very well and had no strict deadline like in Scrum. The respondent who wanted to switch to Scrum mentioned that he thought Scrum fitted better for bigger teams than Kanban.

Out of the 14 respondents using both Scrum and Kanban in their daily work, seven preferred Scrum. The main reason was that it kept everyone involved and informed, thereby improving the transparency of who was doing which task in a team. One mentioned that he particularly liked the stand-up meetings and the sprint reviews (demos). Some said that it fitted the current projects well. Six respondents using both Scrum and Kanban preferred Kanban. Two mentioned that it was because in Kanban there was little overhead and a clear overview, so in Kanban there were bare necessities.

There were 24 respondents using other software development processes. This was a mixed group of respondents using their own process, waterfall, and Agile processes other than Scrum and Kanban. All of the respondents using waterfall would prefer Scrum, and three using their own processes would also like to change to Scrum. The main reasons for switching to Scrum were that it would be easy to work with and could give everyone involved a good overview. Three respondents wanted to use Kanban instead of the process they had been using. They justified that Kanban could fit those tasks for which "what comes next" was an unknown variable, that it was very simple, and that the way the flow of the model worked was desirable.

Overall, although the participants were two groups of different practitioners in software industry, the strengths and weaknesses of Scrum and Kanban identified through the survey and the interviews were similar, such as structuredness, continuous improvement and delivery, and inflexibility of sprints in case of Scrum and simplicity, flexibility, and interruption due to context switching in case of Kanban. In real-life practices there are still a mix of experience, attitude, and acceptance toward Scrum and Kanban. Nonetheless, some practitioners adopted an eclectic approach, probably with the goal of integrating the preferable features of the two processes.

6. DISCUSSION: FACTORS INFLUENCING THE IMPLEMENTATION OF SCRUM AND KANBAN

Results of our empirical studies imply that successful or futile implementations of the highly structured Scrum and the relatively more flexible Kanban depend on certain facilitating and hindering factors. Some interviewees and survey respondents reflected on such factors. We discuss each of them in the following.

6.1. Size Does Matter

The size of a company, a project, and a customer (in terms of financial status) can significantly determine which of the two processes—Scrum or Kanban—is selected. This observation is derived from the data of both the interviews and the survey. Although a big company normally favors using Scrum, the number and complexity of corporate rules typically increases when a company grows. The rules prescribed by Scrum may be incompatible with those of the company. On the other hand, the size of a company, a project, and a customer can be too small for using Scrum when requisite human resources are not available. Two interviewees commented,

> We were much better, when we were smaller, because simply we were being constrained by processes we don't have control over. We are less agile today than we were before the acquisition. But that's a side effect of a big corporation. In that respect we are doing worse. (P1)
>
> They [Agile methods] tend to evolve around larger projects, where you can put up a very effective system with all players included . . . adding a new little feature selling 40 hours this month, then nothing, then another month another 40 hours, some small projects this way. You cannot run them in Scrum. (P4)

6.2. Limited or Late Involvement of Customers

Although some customers appreciate or even demand that they are closely involved in the development process, some may not be willing to get involved after the contract has been signed off. More serious is not involving customers early enough in the process:

> . . . well sometimes when would deliver a kind of big project. We call in the users and sit with them and we do demos, we do demonstration with the users and, but what. The missing link is, as I see it, in the beginning. We don't work closely with the users in the beginning, you know getting the requirements. I think that's the weak part. . . . What my people complain mostly about is, again in the beginning of the process, the lack of communication with the users, and not a clear vision of the products. That we are developing and not a clear vision of, you know, where we are going with the business units. (P8) (NB: here 'users' are employees in the same company of P8 but in different units.)

6.3. Tender Obligation

Changes of requirements are not allowed; this may hinder the Agile process in general but may facilitate Scrum. As remarked by the interviewee P4,

> If we signed a tender, you know they already have done the analysis and requirements analysis. And you have a fixed amount what's supposed to be in the system and you have to deliver it on budget and on time. So, using Agile, you know within that is of course doable, but you know it's not truly agile, because agile means that you can change requirements, you know get the customer with you. Some tenders are from the government so the government doesn't want to participate in that kind of thinking.

6.4. Lack of a "Sellability" Metric

The value-driven continuous delivery is upheld as the Holy Grail for Scrum and Kanban. As listed in Table 1, default metrics like Velocity and Lead Time can be used as monitoring tools. The former can be increased (or the latter can be decreased) at the expense of the quality of user-based activities such as iterative testing. Similarly, the pressure to produce a sellable MMF within a short period may run the risk of adding some demonstrable but not usable or useful features. This can be illustrated by P5's remark:

> Well one obvious value is sellable functionality. And we don't systematically measure these but there is a concept in Lean, it's called Minimum Marketable Features we try to map everything into projects that lead into MMFs. Sometimes a lot of the tasks are just small incremental additions to the systems or you can't really see it in a larger perspective. Yeah. Value, yes. It's an elusive term, I would say.

6.5. Practice in Reality

One intriguing observation shared by the interviewees concerning the usage of Scrum and Kanban in real-life contexts is highly constrained by the corporate culture and structure. Consequently, all methods, irrespective of their prescriptiveness or flexibility, need to be adapted (often by relaxing the rules or guidelines) to address contextual constraints. This observation corroborates with that made by the recent survey on the applications of usability evaluation methods in industry (Følstad, Law, & Hornbæk, 2012). Accordingly, HCI practitioners decompose and recompose different usability evaluation methods in the way that they can apply with ease, comfort, and confidence. The following quotes can well illustrate the points:

> There are always tweaks, I don't think anyone uses pure Scrum, there are always tweaks. And it's a bit of challenge always, we use Agile methods, both in a big company where there are a lot of processes and more waterfall like, also when you are contracting with big customers that have very strict processes on how they procure software and deal with contracts, so it's a bit of a challenge always trying to tweak that reality into some agile ways." (P2)
>
> Based on experience I think Agile has done well, meaning that you can involve the customer and the customer can and will invest the time, that is really really beneficial when it comes to waste. It's very hard to do it right, it's very easy to think you are doing Agile but actually you are doing something very different. And you can say you are doing Scrum or Agile but you are basically not. So there is a big difference between doing Scrum as a process and actually being agile. The difference will show up in your productivity. (P4)

A critical issue not explicitly addressed by the interviewees but vital for UX work is that customers are not surrogate end-users. In all interviews, needs, satisfactions, and concerns of customers were discussed. In some cases such as P4, customers were those who would use software, but in some cases they were not. Given the basic principle of UX work is to understand experiential response of users or people when interacting with an interactive system, the underemphasis of the role of end-users seems an alarming issue. This issue is particularly acute if productivity in terms of continuous delivery is of the highest priority while neglecting the key issue of experiential quality. Furthermore, in critically reflecting on the evolvement of the UCD approach, from its inception in the 1980s through the birth of ISO 13407: 1999 to the recent release of ISO 9241-210: 2010, one can argue that an appropriate way of framing the issue should be the creation of UX-Agile and UX-Lean rather than Agile-UX and Lean-UX to highlight the foremost significance of understanding and addressing user needs in effective as well as efficient software development.

7. IMPLICATIONS AND LIMITATIONS

In this section we first revisit the issues that have driven our research work just presented and then discuss the limitations that influence the generalizability of the findings.

7.1. Revisiting the Research Goal

As mentioned in the Introduction, the main research goal that has driven this work is to examine the issue whether Scrum or Kanban can support UX work better. Results of our analytic and empirical studies imply that the fundamental differences between UX and Agile/Lean in terms of their philosophies, methodologies, and practices make their *full* integration very challenging. Specifically, the key principle of UX of involving real end-users throughout the software development process is hard to be realized in Agile or Lean processes. As indicated by the results of the interviews, both Scrum and Kanban are developer and customer oriented, where "customer" is referred to those who buy services offered by a software company. In comparison, a Kanban team tend to seek users' voices more actively than in a Scrum team, but not to the extent that UX specialists would recommend. This is a compelling concern, as reflected in the title of the paper, *whose experience do we really care about*—customer, user, developer, user, manager, or other stakeholders? Obviously, the answer depends on what "we" is. If "we" are UX specialists, users' needs and preferences are of their prime concern. However, if "we" are product owners and developers, satisfaction of customers' demands may be of higher priority. The practitioners, at least those who were involved in our empirical studies, were rather well informed about the characteristics of these different roles, as shown by their use of ISO definitions to describe the terms *customer* and *user*. Nonetheless, conceptual knowledge may not always be translatable into practices, depending on various contextual constraints.

One critical fact repeatedly and explicitly pointed out by the participants of our studies is that resources for UX work are often limited in Agile software projects. This is consistent with what has been reported in previous studies (e.g., Da Silva et al., 2013; Illmensee & Muff, 2009; Salah et al., 2014; Wale-Kolade, 2015). Consequently, companies tend to perform evaluations informally with only few users (who may simply be colleagues sitting in neighboring cubicles in an open office) collecting feedback in ad hoc manner.

Another intriguing implication inferred from the data is loose adherence to principles and guidelines. Many of the practitioners adopted an eclectic approach, combining different techniques and tools of Scrum, Kanban, and some other Agile processes to create tailor-made in-house software development processes to fit the style and ethos of the team. This kind of deconstruction and reconstruction of existing methods in real-life practice has been examined in the context of usability evaluation (e.g., Følstad et al., 2012; Woolrych, Hornbæk, Frøkjær, & Cockton, 2011), and such mix-and-match approaches are regarded as practical and even necessary.

7.2. Limitations

We are aware of some limitations of our current work. First, our participants are limited to the practitioners from the software industry in Iceland. According to Eurostat,[3] Iceland was one of the four European countries (Sweden, Finland, and Ireland) that had the largest share (> 4%) in total employment of both manufacturing and services in the high-technology sectors in 2013. Presumably, this status remains valid. For a small country with the population of about 300,000 people, it is a rather significant figure. As the high-tech sectors keep abreast with the new methods of software development processes, the professionals therein should have relevant experience and knowledge in Agile and Lean methods. Although we are well aware that a larger sample from different countries could provide more insights into the issues, the data of the current studies have laid an initial and important groundwork. As the adoption of Kanban is a more recent trend, currently the number of case studies is relatively small. It is anticipated it will increase in the coming years.

Another aspect of the current studies we could have improved is the use of secondary data, given that the semistructured interviews were aimed at studying wastes of software development. In fact, the creation of Agile methods was partly motivated by the goal of minimizing such wastes. Nevertheless, the questions posed to the interviewees are highly relevant to our research interest, especially those related to the process of requirements engineering. In our future work,

[3] http://ec.europa.eu/eurostat/statistics-explained/index.php/High-tech_statistics

specific UX questions will be presented to interviewees on how Agile/Lean development teams address the issue of aesthetic quality in software and of the temporal changes of users' experiences.

Furthermore, as repeatedly mentioned by the participants of our studies, corporate culture and structure have a restraining effect on the application of Scrum and Kanban. Although conducting empirical studies to investigate this phenomenon systematically is beyond the scope of this article, we consider it as an item to be high in the future research agenda on successful integration of UX and Agile/Lean work.

Customer experience (CX; e.g., Meyer & Schwager, 2007; Novak, Hoffman, & Yung, 2000; Schmitt, 2003) is not the focus of the current study, although our participants might care more about CX than UX, as suggested by their frequent references to customers. As shown in Figure 1, one part of CX overlaps with UCD/UX and the other part is beyond the HCI boundary, addressing more on marketing and management. This raises a recurrent topic about the paradoxical relations between the two fields—HCI and Management Information Systems, which are close but tend to keep their status distinct from each other. Nonetheless, a deeper analysis of CX–UX relationship can be considered as a new line of inquiry calling for future research work.

8. CONCLUSION

It is an exciting time to witness the recent development in the field of HCI and software engineering with the introduction of new hybrids such as UX Lean and UX Agile. From the academic perspective, researchers are motivated to understand the underpinning philosophies, theoretical frameworks, and methodological assumptions. These backdrops allow us to explain as well as predict the success and failure of applying such emerging approaches. From the industrial perspective, practitioners are keen to know which approach works effectively and efficiently in terms of releasing the highest possible quality in the shortest possible time within a lowest possible budget. These criteria, unfortunately, oftentimes are contradicting. Although our conceptual analysis of Scrum and Kanban suggests that the flexibility of Kanban seemed to provide a better fit for UX, the results of our empirical studies were ambivalent. Neither of the emerging software development processes supports UX effectively, given the intriguing observation that users are referenced significantly less often than are customers. Findings of the follow-up survey indicate that the practitioners involved seem not care much about the distinction between the two concepts. This is a compelling concern that necessitates more research work to gain deeper insights into the prevalence of the issue and to identify strategies to address it. Overall, we aim to sustain our research effort in tackling the challenge of understanding how both fields—HCI and software engineering—are best integrated to serve people's needs.

REFERENCES

Bargas-Avila, J. A., & Hornbæk, K. (2011). Old wine in new bottles or novel challenges: a critical analysis of empirical studies of user experience. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2689–2698.

Beck, K., Beedle, M., van Bennekum, A. Cockburn, A., Cunningham, W., Fowler, M., . . . Jeffries, R. (2001). *The agile manifesto*. & The Agile Alliance. Retrieved from http://www.agilealliance.org/the-alliance/the-agile-manifesto/

Beyer, H. (2010). User-centered agile methods. *Synthesis Lectures on Human-Centered Informatics, 3*, 1–71.

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, *70*, 213.

Da Silva, T., Silveira, M. S., & Maurer, F. (2013). Ten lessons learned from integrating interaction design and agile development. *Agile Conference*, 42–49.

Da Silva, T. S., Silveira, M. S., Maurer, F., & Hellmann, T. (2012). User experience design and agile development: From theory to practice. *Journal of Software Engineering and Applications, 5*, 743.

Diebold, P., & Dahlem, M. (2014). Agile practices in practice: A mapping study. *Proceedings of Evaluation and Assessment in Software Engineering*, Article no. 30.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N.B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software, 85*, 1213–1221.

Fagerholm, F., & Pagels, M. (2014). Examining the structure of lean and agile values among software developers. *Agile Processes in Software Engineering and Extreme Programming*, 218–233.

Ferreira, J., Sharp, H., & Robinson, H. (2010). Values and assumptions shaping agile development and user experience design in practice. *Agile Processes in Software Engineering and Extreme Programming*, 178–183.

Fleiss, J. L., Levin, B., & Paik, M. C. (1981). The measure of interrater agreement. In *Statistical Methods for Rates and Proportions* 2nd ed. (p. 212–236). New York: John Wiley & Sons.

Følstad, A., Law, E., & Hornbæk, K. (2012, May). Analysis in practical usability evaluation: a survey study. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2127–2136.

Gentile, C., Spiller, N., & Noci, G. (2007). How to sustain the customer experience: An overview of experience components that co-create value with the customer. *European Management Journal, 25*, 395–410.

Gosper, J., & Binnie, A. (2011). UX design and agile: A natural fit? *Communications of the ACM, 54*, 54–60.

Gothelf, J. (2013). *Lean UX: Applying lean principles to improve user experience*. Cambridge, MA: O'Reilly Media.

Hartmann, J., Sutcliffe, A., & de Angeli, A. (2008). Towards a theory of user judgment of aesthetics and user interface quality. *Transactions on Computer Human Interaction, 15*(4).

Hassenzahl, M. (2004). The interplay of beauty, goodness, and usability in interactive products. *Human–Computer Interaction, 19*, 319–349.

Hocko, J. (2011). User-centered design in procured software implementations. *Journal of Usability Studies, 6*, 60–74.

Hofman, R. (2011). Behavioral economics in software quality engineering. *Empirical Software Engineering, 16*, 278–293.

Ikonen, M., Pirinen, E., Fagerholm, F., Kettunen, P., & Abrahamsson, P. (2011). On the impact of kanban on software project work: An empirical case study investigation. *2011 16th IEEE International Conference on Engineering of Complex Computer Systems* (ICECCS), 305–314.

Illmensee, T., & Muff, A. (2009). 5 users every Friday: A case study in applied research. *Agile Conference*, *2009*, 404–409.

ISO/IEC 12207:2008. Systems and software engineering—Software life cycle processes.

ISO 9241-210:2010. Ergonomics of human-system interaction—Part 210: Human-centred design for interactive systems.

Karapanos, E. (2013). User experience over time. In *Modeling users' experiences with interactive systems* (pp. 57–83). Berlin, Germany: Springer.

Kniberg, H., & Skarin, M. (2010). *Kanban and Scrum: Making the most of both*. C4media. Retrieved from http://www.infoq.com/minibooks/kanban-scrum-minibook

Krippendorff, K. (2004). *Content analysis: An introduction to its methodology* (2nd Ed.). Sage.
Kuusinen, K., Mikkonen, T., & Pakarinen, S. (2012). Agile user experience development in large software organization: Good expertise but limited impact. *Proceedings of Human-Centred Software Engineering*, 94–111.
Lárusdóttir, M., Cajander, Å., & Gulliksen, J. (2014). Informal feedback rather than performance measurements–user-centred evaluation in Scrum projects. *Behaviour & Information Technology*, *33*, 1118–1135.
Lárusdóttir, M. K., Haraldsdottir, O., & Mikkelsen, B. (2009). User involvement in Icelandic software industry. In *Proceedings of the 2nd International workshop on the Interplay between Usability Evaluation and Software Development*, (pp. 51–52).
Lavie, T., & Tractinsky, N. (2004). Assessing dimensions of perceived visual aesthetics of web sites. *International Journal of Human-Computer Studies, 60*, 269–298.
Law, E. L. C., Roto, V., Hassenzahl, M., Vermeeren, A. P., & Kort, J. (2009, April). Understanding, scoping and defining user experience: A survey approach. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 719–728.
Law, E. L. C., & van Schaik, P. (2010). Modelling user experience—An agenda for research and practice. *Interacting with computers, 22*, 313–322.
Law, E. L. C., van Schaik, P., & Roto, V. (2014). Attitudes towards user experience (UX) measurement. *International Journal of Human-Computer Studies, 72*, 526–541.
Leszek, A., & Courage, C. (2008, August). The doctor is" in"—Using the office hours concept to make limited resources most effective. *Agile Conference 2008*, 196–201.
McCarthy, J., & Wright, P. (2004). *Technology as experience*. Cambridge, MA: MIT Press.
Meyer, C., & Schwager, A. (2007). Understanding customer experience. *Harvard Business Review, 85*, 116.
Miller, L., & Sy, D. (2009). Agile user experience SIG. *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, 2751–2754.
Modig, N., & Åhlström, P. (2012). *This is Lean—Resolving the efficiency paradox*. Halmstad: Bulls Graphics AB.
Novak, T. P., Hoffman, D. L., & Yung, Y. F. (2000). Measuring the customer experience in online environments: A structural modeling approach. *Marketing Science, 19*, 22–42.
Ohno, T. (1988). *The Toyota production system: Beyond large-scale production*. London, UK: Productivity Press.
Olson, G. M., & Olson, J. S. (2003). Human-computer interaction: Psychological aspects of the human use of computing. *Annual Review of Psychology, 54*, 491–516.
Poppendieck, M., & Poppendieck, T. (2007). *Implementing lean software development: From concept to cash* (3rd ed.). New York, NY: Addison-Wesley Professional.
Reja, U., Manfreda, K. L., Hlebec, V., & Vehovar, V. (2003). Open-ended vs. close-ended questions in web questionnaires. *Developments in Applied Statistics, 19*, 159–177.
Salah, D., Paige, R. F., & Cairns, P. (2014). A systematic literature review for agile development processes and user centred design integration. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, (5–14).
Schmitt, B. H. (2003). *Customer experience management: A revolutionary approach to connecting with your customers*. New York, NY: Wiley & Sons.
Schwaber, K. (1997). Scrum development process. In *Business Object Design and Implementation* (pp. 117–134). London, UK: Springer.
Schwaber, K. (2009). *Agile project management with Scrum*. Microsoft Press.
Schwaber, K., & Beedle, M (2002). *Agile software development with Scrum*. Upper Saddle River, NJ: Prentice Hall.
Seffah, A., Vanderdonckt, J., & Desmarais, M. C. (Eds.). (2009). *Human-centered software engineering: Software engineering models, patterns and architectures for HCI*. London, UK: Springer.
Shingo, S. (1982). *Study of Toyota production system from an industrial engineering viewpoint*. New York, NY: Productivity Press.
Singh, M. (2008). U-SCRUM: An agile methodology for promoting usability. *Proceedings of the IEEE AGILE Conference 2008*, 555–560.
Sohaib, O., & Khan, K. (2010, June). Integrating usability engineering and agile software development: A literature review. *2010 international conference on Computer design and applications*, V2–32.
Ungar, J., & White, J. (2008). Agile user centered design: Enter the design studio-a case study. *CHI'08 Extended Abstracts on Human Factors in Computing Systems*, 2167–2178.
Van Dyne, L., & Pierce, J. L. (2004). Psychological ownership and feelings of possession: Three field studies predicting employee attitudes and organizational citizenship behavior. *Journal of Organizational Behavior, 25*, 439–459.
Vartanian, T. P. (2010). *Secondary data analysis*. Oxford, UK: Oxford University Press.
Vermeeren, A. P., Law, E. L. C., Roto, V., Obrist, M., Hoonhout, J., & Väänänen-Vainio-Mattila, K. (2010, October). User experience evaluation methods: Current state and development needs. *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, 521–530.
VersionOne. (2014). *State of Agile survey*. Retrieved from http://stateofagile.versionone.com
Wale-Kolade, A. Y. (2015). Integrating usability work into a large inter-organisational agile development project: Tactics developed by usability designers. *Journal of Systems and Software*, *100*, 54–66.
Woolrych, A., Hornbæk, K., Frøkjær, E., & Cockton, G. (2011). Ingredients and meals rather than recipes: A proposal for research that does not treat usability evaluation methods as indivisible wholes. *International Journal of Human-Computer Interaction, 27*, 940–970.
Wright, P., & McCarthy, J. (2010). Experience-centered design: Designers, users, and communities in Dialogue. *Synthesis Lectures on Human-Centered Informatics*. Morgan & Claypool. Retrieved from http://www.morganclaypool.com/doi/abs/10.2200/s00229ed1v01y201003hci009?journalCode=hci

ABOUT THE AUTHORS

Effie Lai-Chong Law is Reader at University of Leicester, UK. Her research focuses are usability and User Experience (UX) methodologies and their applications in various domains, including technology-enhanced learning, serious games in the context of law and cultural heritage, and cross-cultural User-Centred Design (UCD) research. She has been involved in a number of EU- and UK-funded projects, leading the UCD work in them.

Marta Kristín Lárusdóttir is an assistant professor at Reykjavik University, Iceland. Her main research area is the interplay between usability and UX activities and software development processes, especially in Agile development processes such as Scrum and Lean software development. She has been collaborating with various software companies in industry, and also an active member in IFIP TC13.