# Chapter 9
# BoB: A Framework for Organizing Within-Iteration UX Work in Agile Development

**Kati Kuusinen**

**Abstract** Most research on Agile UCD recommends scheduling of UX work one iteration ahead of development. There is, however, some evidence arguing for an approach where software developers and UX specialists work in cross-functional teams conducting design and implementation tasks during the present iteration. This within-iteration approach can, for instance, improve communication between UX designers and software developers and thus help the team to better concentrate on value-adding work. This chapter discusses problems related to the iteration-ahead approach and introduces a framework called BoB (Best of Both Worlds) that utilizes the within-iteration approach to integrate UX work in agile development. Furthermore, we present guidelines related to factors that support the within-iteration approach and the cross-functional team.

**Keywords** User experience (UX) • Agile development • User-centered design (UCD) • UX design work • Agile UX • Human-computer interaction (HCI)

## 9.1 Introduction

Good user experience (UX) can be a significant competitive advantage in the market for a software product e.g. [1, 2]. However, good UX should not be a game of chance, but it requires deliberate work. By the phrase *UX work*, we refer to activities that aim at developing software that is usable, fulfills user needs, and provides desired UX. Furthermore, *UX design* defines how users interact with, and react to, software In many cases, designing for user interaction and UX requires special skills that are beyond the skills that software developers are expected to possess. Thus, a role of a UX specialist (UXS) is often needed to ensure fluent user flow and desired UX from use of the software under development. However, current agile development methodologies provide no guidance on how to include a UX specialist

K. Kuusinen (✉)
Tampere University of Technology, Tampere, Finland
e-mail: kati.kuusinen@alumni.aalto.fi

role in a project, nor do they guide organizing collaboration between UX specialists and software developers.

UX work has traditionally relied on heavy upfront studies and careful design before starting the implementation. Although the UCD process produces design solutions in repeated phases of design and evaluation with users, such iterations are not compatible with agile development iterations. Researchers and practitioners have been developing more compatible practices since agile methodologies started becoming popular. According to recent systematic literature reviews [3, 4], academic research most frequently recommends utilizing the *iteration-ahead* approach originated by Miller [5] and Sy [6]. Here, UX work is conducted one to two iterations ahead of development according to the time requirement of, for instance, user research or UX design activities. Thus, there are distinct sequential phases for user studies, design work, and implementation. In addition, most of the related research recommends conducting an upfront design phase before starting implementation [3, 4]. Furthermore, most UX work is conducted by UX specialists [3]. These practices are against the most popular agile principles (e.g, Scrum) in that they necessitate pre-planning before each development iteration and divide design and development activities among distinct persons.

In this chapter, we introduce BoB framework as an alternative to the commonly recommended iteration-ahead approach for integrating UX work in agile development. In our framework, UX design and even some lightweight user studies are conducted together with development activities in the same iteration, i.e. it utilizes the within-iteration approach. We have named the framework BoB. The name comes from *Best of Both worlds*, as it combines such advantages from UCD and agile development that normally are considered to be mutually exclusive, i.e. it is expected that one cannot have both at the same time. The naming was influenced by the television series *Star Trek: The Next Generation* episode with the same name [7]. Our approach aims to tackle several challenges connected to the iteration-ahead approach, for instance, related to communication and timing of the work. BoB builds on the practices of conducting development work iteratively, and having a cross-functional team including both software development and UX specialist competences.

We have developed BoB based on empirical research on agile UX work in nine software companies based in Finland over the years 2011–2015. Most of the research has been conducted on organizations and projects developing enterprise software and work-related tools. The contributing research is presented in [8–16]. Although features of BoB have been used in the studied companies, the framework as such has not been fully used in none of them.

The rest of this chapter is structured as follows: Section 9.2 introduces the iteration-ahead approach and describes problems related to it. Section 9.3 discusses the concept of a cross-functional team. In Sect. 9.4, we introduce factors that we find supporting the within-iteration approach, and in Sect. 9.5, we present the BoB framework for organizing UX work when using a cross-functional team and the within-iteration approach. Section 9.6 discusses the introduced framework. Finally, Sect. 9.7 presents closing remarks for the chapter.

## 9.2 Iteration-Ahead Approach

### *9.2.1 Overview*

Sy [6] introduced the "*one sprint ahead*" approach (Fig. 9.1) in 2007. Since then, it has become a popular means to integrate UX work in agile development [3]. When following the approach, development starts with *iteration 0,* during which upfront planning and studies are conducted. Most research in agile UCD suggests doing some upfront planning but also keeping it to a minimum since heavy design upfront is against agile principles [3]. After iteration 0, development is divided to two separate tracks, one of them concentrating on technical implementation (the upper track in Fig. 9.1) and the other concentrating on UX work. Developers start by building features that have less impact on the UX, such as certain backend solutions in the first actual iteration. At the same time, UX specialists design for features that will be implemented in iteration 2 and study for features that will be implemented in iteration 3. Following iterations are conducted similarly; UX specialists design one and study two iterations ahead of development. They also conduct user tests on the functionality implemented during the preceding iteration.

The approach was developed at the Autodesk company to adopt agile practices within UX work after utilizing waterfall development for years. The company had a functional and proficient UX team with established practices before agile adoption [6]. Thus, the company culture was presumably already supporting UX work. In contrast, agile development was new to the company. Sy [6] reported that the "one sprint ahead" approach was introduced to tackle problems, such as excessive design inventory, outdated design, and lack of common vision in the project. Design inventory is produced when UX is designed ahead of development, such as in waterfall development, and ready-made design has to wait to be implemented. If,

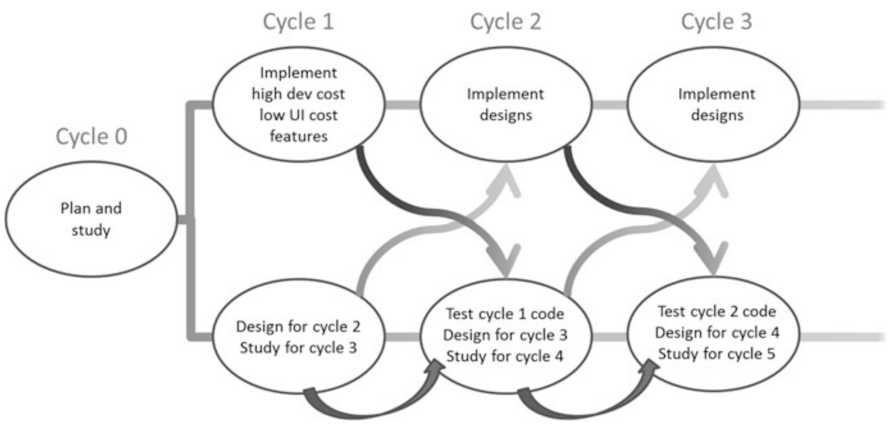

**Fig. 9.1** The "one sprint ahead" approach [6]

for instance, requirements change during the waiting time, the design may become outdated and require revision. Indeed, compared to waterfall development, the one sprint ahead approach surely reduces the size of design inventory since only a small amount of design is waiting to be developed at a time. Moreover, as the design is produced just in time, it should be up to date. There are, however, some challenging issues in the iteration-ahead approach that we will discuss next.

### *9.2.2 Challenges in the Iteration-Ahead Approach*

Utilizing the one iteration-ahead approach has its drawbacks. As it consists of successive design and development phases, it easily becomes a mini-waterfall where development proceeds in small phases of studying, designing, implementing, and testing. The iteration-ahead approach practically divides development work into two separate tracks yielding that technical implementation is conducted on one track and user studies, UX design, and user testing are conducted on another track that is scheduled to serve the implementation pace.

Most of the challenges in using the iteration-ahead approach are related to the fact that utilizing the model necessitates pre-planning before and testing after each development iteration. For instance, in 2-week implementation cycles, developers need to know 2–4 weeks in advance what they will be implementing whereas test results will be available 2 weeks after the development. Thus, the iteration-ahead approach makes the feedback loop grow from 2 to 6 or 8 weeks which makes responding to change slower and more difficult. Although the idea of the iteration-ahead approach is to get continuous user feedback, the actual developed artefacts, i.e. the working software, can be tested with users only after three to four iterations instead of one.

A major issue that agile methodologies were developed for is that in software engineering, one cannot know in advance what the system under development should be like and how it should be implemented [17]. Thus, a fundamental principle of agile methodologies is to welcome late change [18, 19]. When there is a change in the development order, UX design work cannot adapt to that if UX specialists need, for instance, 2 weeks to study and another 2 weeks to design. The situation is illustrated in Fig. 9.2, where in the fourth iteration, priorities on the backlog change and developers start building functionality C instead of functionality B, which they were originally planning to implement next. Therefore, there is no UX design for functionality C available. In such situations, UX designers start to hurry the design and developers need to improvise [8]. We have observed that although the UX suffers in this case, it will rarely be improved later [8, 9].

Another drawback in the approach is that UX specialists test the implementation in the next iteration while developers are already writing new code based on the current implementation. In practice, this usually means that implementation cannot be iterated based on the findings—it would be too costly and require rework [8, 9]. This is illustrated in Fig. 9.2 where functionality A fails the usability test conducted

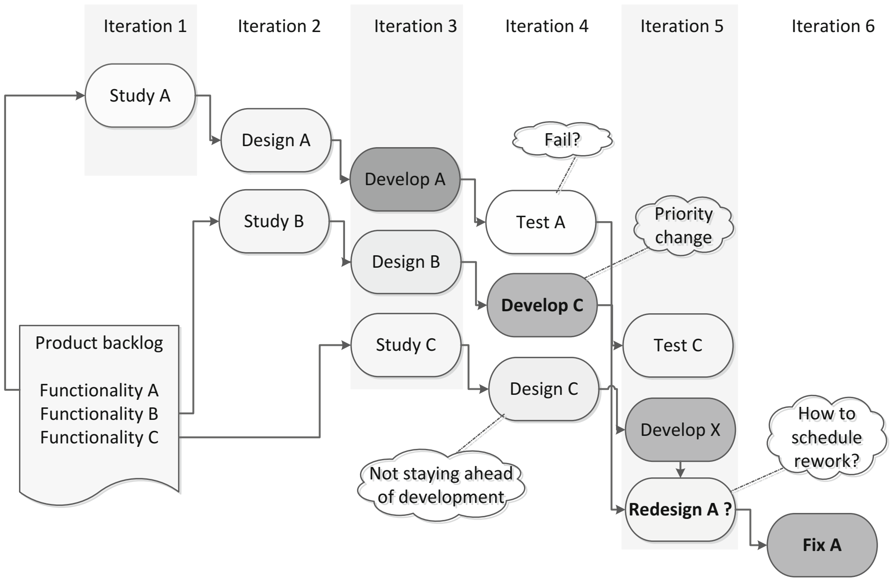

**Fig. 9.2** Waterfall characteristics of the iteration-ahead approach. Changes in backlog priority order or failing a user study are problematic when utilizing the approach

in Iteration 4. If one fixes Functionality A, it will be redesigned in Iteration 5 and re-implemented in Iteration 6. To avoid building on erroneous code, developers need to be working with something that is not influenced by Functionality A. In practice, this necessitates UX design inventory, or that the developers can focus on something that does not require UX design work.

Both aforementioned issues make it more difficult to keep the UX design work iteration ahead of development. Moreover, both scenarios are common in agile development. For instance, in Scrum, the task list is re-prioritized before each iteration [20], and user testing would be unnecessary if it did not reveal defects. One of the biggest challenges in agile UX work is to synchronize the UX specialists' and developers' work [21]. In addition, although Sy [6] emphasized continuous communication between developers and UX specialists, communication problems in utilizing the approach are common [21, 22]. Communication between disciplines is generally more challenging than within a discipline [23]. Also, within-team communication is more efficient than communication with people outside the team [22]. The third issue we have noticed that makes the communication more difficult is the temporal difference between the mindsets of developers and UX designers [15]. When a UX designer asks developers to evaluate certain decisions, the designer is planning to consider a feature the developers will be implementing in the next iteration; however, the developers cannot really relate to it. They have not been thinking about that particular feature yet, and thus it is hard for them to make good decisions considering it. Thus, their opinion is likely to change as they start implementing that particular feature.

Finally, the iteration-ahead approach separates front-end design to an iteration 0, which can be longer than the actual development iterations; however, "*they occur in weeks rather than months*" [6]. The idea of iteration 0 is to arrange time for planning and user data gathering [6]. However, separating design work into an upfront design phase often means that agile practices are used only after the upfront design [6, 8]. Thus, iteration 0 is often conducted with non-agile project management practices and the rest of the project is managed with agile practices. This leads to double project management and to having an avoidable barrier between design and development [8].

## 9.3 Cross-Functional Team Vs. Separate UX and Development Teams

Ideally, an agile team should include all the expertise necessary to define, design, build, and test running software that satisfies customer needs [19]. A team with all the required expertise is cross-functional. However, agile UCD practices rarely include a UX specialist in the development team [3]. Instead, the UX specialist often works outside of the team, which easily hinders collaboration [8, 22] and thus does not make the best use of UX expertise. Attempts to improve the situation have been made. For example, "*dual track Scrum*" [24, 25] is an approach similar to the one sprint ahead approach in that it divides product discovery and implementation into separate tracks. In dual track Scrum, however, a cross-functional team instead of UX specialists only is responsible for the second track. The team–typically consisting of a product owner, UX specialist and a developer–works towards refining and validating product backlog items for the forthcoming development iterations [26]. Unfortunately, we could identify no research articles on dual track Scrum and thus cannot build on any formal studies of its advantages and disadvantages.

Ferreira et al. [22] found that organizational values have an impact on how it is beneficial to integrate UX work with agile development: some organizations value separate UX and development work, while others value togetherness. Cross-functional team approaches for integrating UCD with agile development practices can thus be especially beneficial for organizations valuing togetherness. Ferreira et al. [22] studied two projects, one of them utilizing a separatist approach in which a UX specialist directed developers' work—basically utilizing the iteration-ahead approach—and the other working in a cross-functional team within the iteration sharing the power of decision. They [22] observed that communication was more efficient in the cross-functional team and the team did not need such activities as interpreting the design as it did in the separatist approach.

Including people from different disciplines makes communication generally more challenging [23], but separating UX specialists into their own teams, instead of including them in agile development teams, may easily lead to degraded communication. In such situations, developers tend not to take ownership of UX issues and UX specialists become seen as outsiders. When UX specialists and developers are separated, teams encounter problems with timing and the implementability of the design [8, 27]. Moreover, in those settings where a separate UX team delivers the UX design, the development team needs additional coordination activities to interpret the design, to identify mismatches between the new design and existing code, and to determine what is already implemented [22]. Several authors discuss the role of UX specialists compared to development teams. Isomursu et al. [28] concluded that UX specialists' responsibilities should be in line with the expectations of development teams. Hodgetts [29] considered it vitally important for UX practitioners to see themselves as part of a project team and to conduct their tasks according to that perception. Lee [30] stated that UX specialists need to be active participants in order to be embedded in agile teams. Finally, Kuusinen et al. [8] found that including UX specialists in development teams was the preferred approach amongst practitioners for integrating UX work with agile development practices.

Thus, while much related research argues for separation of UX design and development activities [3], it is not an approach that is suitable in all cases. Moreover, to our knowledge, there are no earlier frameworks in related research for organizing agile UX work in cross-functional teams working within-iteration.

## 9.4 Method

Our research goal was to develop a construct for integrating UX work with agile practices in the context of enterprise software development. Our research question was as follows: *Which activities support the integration of agile development and UX work?* To support answering the main research question, we studied also tasks and goals that comprise agile UX work, and the challenges that companies encounter while integrating UX work with agile development.

We selected a 'building theories from case studies' research strategy [31] for developing the construct (Table 9.1). It is a "*research strategy that involves using one or more cases to create theoretical constructs, propositions, and/or midrange theory from case-based, empirical evidence*" [32].

A case study is "*an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident.*" Previous research guides data collection and analysis in case studies, and both multiple sources of evidence and research methods are utilized ([33], p. 13).

We selected the research strategy for the following reasons: Firstly, case studies are a common research methodology when studying software engineering [34]. We had very limited ability to control or affect the studied phenomenon, and the phenomenon was not well-known before. Secondly, our aim was at building

**Table 9.1** Process of building theory from case study research [31]

| Step | Activity |
|---|---|
| Getting started | Definition of research question |
| Selecting cases | Theoretical, not random sampling |
| Crafting instruments and protocols | Multiple data collection methods, qualitative and quantitative |
| Entering the field | Overlap data collection and analysis |
| Analyzing data | Within-case analysis, Cross-case pattern search using divergent techniques |
| Shaping hypotheses | Iterative tabulation of evidence, Replication, Search evidence for "why" behind relationships |
| Enfolding literature | Comparison with literature |
| Reaching closure | Theoretical saturation when possible |

a construct for explaining and supporting the phenomenon. Thus, we consider Eisenhardt's [31] strategy appropriate for our research.

### *9.4.1 Research Process*

Our research consisted of four rounds of empirical research and of activities that aimed to build a construct (BoB) based on the research (Fig. 9.3). First, we conducted explorative case studies with surveys and interviews in three companies concentrating on challenges organizations encounter in their agile UX work and practices they find beneficial for the work [8, 9, 13]. Having developed an initial understanding here, we next conducted a literature review. Based on the first round research results and the literature review, we planned a second round of more structured studies concentrating on the actual contributing tasks and roles present in UX work in agile projects [10, 12]. We based the study on six development projects in five companies with repeated surveys and interviews. In the third round of research, we included the user perspective in our studies by surveying the perceptions that development teams and users have towards the UX of the software systems the teams have been developing [16]. The third round of research was conducted in the same six projects as the second round. Most of the framework was built after the third round. Finally, we conducted a fourth round of research by interviewing practitioners to evaluate the built framework.

We started to build the BoB framework after the first round of research. We presented the first version of the framework draft in [11]. Most of the framework shaping was conducted after the three first rounds of research in 2014 and 2015. Thus, shaping BoB was based on the three first rounds of research. During year 2015 we compared the framework to practices described in related research as following guidance from Eisenhardt [31] (phase: literature contextualization (called "enfolding literature" in [31])). Finally, in the fourth round of research, we discussed

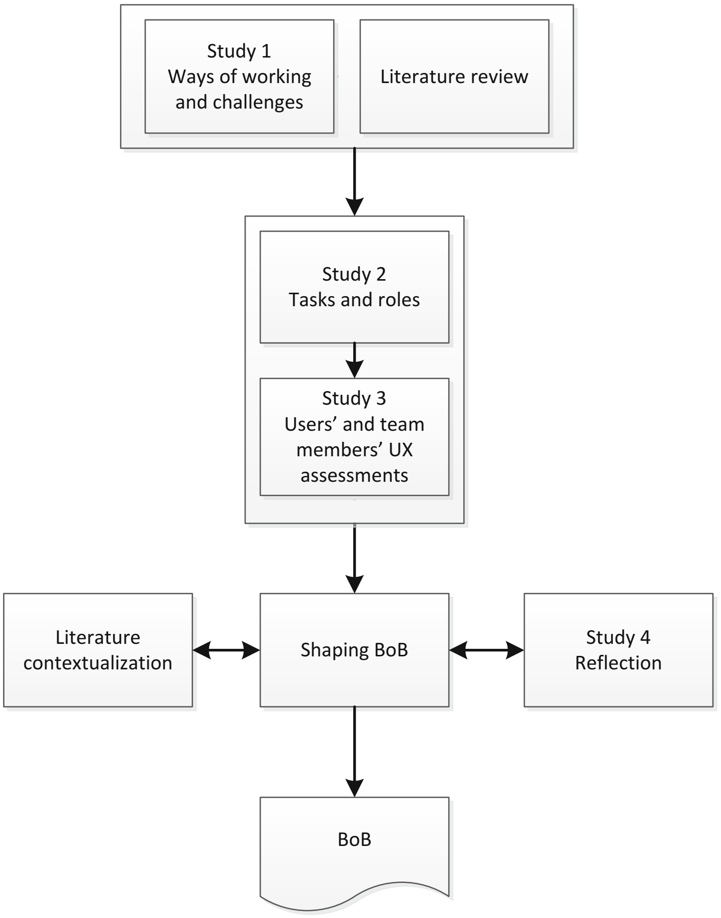
Study 1
Ways of working and challenges
Literature review
Study 2
Tasks and roles
Study 3
Users' and team members' UX assessments
Literature contextualization
Shaping BoB
Study 4
Reflection
BoB

**Fig. 9.3** Research process for building the BoB model

the framework and its viability with UX specialists and a developer from three companies in an interview study to evaluate BoB's practical validity. The research was conducted between years 2011 and 2015. We discuss research methods and limitations in more detail in the original publications.

### *9.4.2 Participants*

Participants and studied companies are introduced in the original publications; we give here only an overview of them. Altogether, we conducted 75 interviews and received 282 survey responses from nine companies (Table 9.2).

**Table 9.2** Summary of methods and participants per study

| Study | Method | Participant roles | Countries | N |
|---|---|---|---|---|
| S I | Survey 1 | Dev, PO, UXS, Arc, Manager, SM,, other (e.g. tester, user support) | Finland 58.0 %, Other 25.87 % (mainly France, Sweden, Czech Republic and Malaysia), Unknown 16.1 % | 143 |
| | Interviews 1 | Dev, PO, SM, UXS,, Arc, Manager, Tester, Customer | Finland 95.2 %, Sweden 2.4 % and China 2.4 % | 50 |
| | Survey 2 | Dev, PO, UXS, Arc | Finland 100 % | 8 |
| | Interviews 2 | Dev, PO, UXS, Arc | Finland 100 % | 7 |
| S II | Pilot interviews | Dev, UXS, Arc | Finland 50.0 %, China 25.0 %, Belarus 12.5 %, India 12.5 % | 8 |
| | Pilot survey | Dev, PO, UXS, SM, Tester | Finland 36.8 %, China 26.3 %, Belarus 26.3 %, India 10.5 % | 19 |
| | Long-term survey | Dev, PO, UXS | Finland 45.2 %, Russia 25.8 %, China 22.6 %, Latvia 3.2 %, Estonia 3.2 % | 31 |
| | Retrospect survey | Dev, PO, UXS | Finland 53.8 %, Russia 30.8 %, China 26.9 % | 26 |
| | Interviews | PO, UXS | Finland 100 % | 6 |
| S III | Team survey | Dev, PO, UXS | Finland 73.1 %, Russia 19.2 %, Latvia 3.8 %, China 3.8 % | 26 |
| | User survey | User | Majority from Finland | 29 |
| S IV | Interviews | Dev, UXS | Finland 100 % | 4 |

Legend: *Arc* Architect, *Dev* Developer, *PO* Product Owner, *SM* Scrum Master, *UXS* UX specialist

Some participants were interviewed or surveyed more than once. Since we did not identify the respondents of the first round surveys, and one of the companies participated in three first rounds of research, we do not know the number of individual participants. Moreover, we did not ask in which country the user participants of the third round of research were based. However, the whole user base of two participating projects was in Finland.

All nine participating companies were developing enterprise software or work-related tools. Five were IT service companies, three were engineering and technology companies, and one was developing its own specialized systems for several platforms. Three companies focused on developing mobile enterprise applications, one developed wireless industrial systems, one developed large industrial safety-critical software-hardware systems, one developed embedded systems for specialist users, one developed tools for both business and consumer users, and two companies developed basically whatever software customers order. Two of the companies were large with around 20,000 employees, two employed 1000–2000 persons, four had 100–500 persons, and one was small with 10–30 employees. Six of the companies were operating globally while three had sites only in Finland. All but one company employed UX specialists. All the companies utilized agile practices at least in some of their organizations.

## 9.5 BoB Framework for Organizing Within-Iteration Agile UX Work

In the resulting BoB framework, developers and UX specialist(s) form a cross-functional team. Work towards certain functionalities is conducted in a cross-functional team within a single iteration. Work can be chunked in several iterations when necessary; however, an iteration should always include both analysis and building activities. In the case of UCD, it means that each iteration should contain activities for understanding the user's needs, designing and developing towards them, and evaluating the appropriateness of the result [35]. Thus, we consider the approach more compatible with agile development.

### *9.5.1 Guidelines to Support the Cross-Functional Team*

We organize our guidelines to support the cross-functional team and the within-iteration approach into factors related to *people*, *process, tasks, tools,* and the *developed artefact* (Table 9.3). We base the taxonomy on the categorization of [3, 36]. Brhel et al. [3] classify agile UCD integration types into *process, people, practices,* and *technology*. Chow et al. [36] classified critical success factors of agile software development as *organizational, people, process, technical,* and *project* factors. Instead of UCD *practices* and supporting *technologies*, we concentrate on *tasks* related to UX work and supporting *tools* in general, respectively. Finally, we consider factors related to the *artefact* developed in a project context. This fifth group of *artefact* factors are omitted from Table 9.3, but are only discussed in the text as we do not want to give actual guidelines on the software under development. Instead, we discuss the impact on the type and characteristics of the software for the within-iteration approach.

Furthermore, agile principles require a motivated cooperative team to deliver, from early on and continuously, software that satisfies the customer [18]. We build our guidelines both on agile principles and a user-focused mindset.

**People Factors** First of all, the team must value togetherness as defined in [22]; team members need to be willing to work together. They should have a positive and curious attitude towards the disciplines of other team members. Cooperation gets easier if team members understand the work of other team members to some extent. For instance, Gulliksen et al. [37] suggest that UX specialists should have some knowledge of software development to improve the communication. Working in a

**Table 9.3** Supporting guidelines for the within-iteration approach

| People | Process | Tasks | Tools |
|---|---|---|---|
| Learn from others: Broaden your competence areas. | Work within one iteration. | Integrate UX work via tasks not via roles. | Communicate UX tasks via backlog. |
| Be willing to cooperate. | Produce working design. | Minimize user studies. | Establish feedback channels. |
| Respect people from other disciplines. | Allow trial and error: accept design debt. | Treat UX-related tasks similarly to other development tasks. | Utilize technologies that allow rapid design and development. |
| Involve the whole team in user communication. | Hurry to markets to enable actual user feedback. | Appreciate professionalism when allocating tasks. | Allow maturity difference in visual and functional readiness. |

cross-functional team becomes a virtuous circle as team members learn from each other [10]. Learning eases task allocation as competence areas of team members become partially overlapping [10].

Like in agile development in general, the team benefits from close collaboration, the emphasis being on face-to-face communication. At the minimum, either the PO or developers should be co-located with the UX specialist [10]. Regarding the big picture of the project and being able to satisfy the user need, the whole team should be involved in communication with users [12]. This is also in line with the principles of UCD. Continuous communication with the user and the customer is especially important in rapid development where all the required information should be at hand whenever needed.

**Task Factors** When a UX specialist is working with the development team and the team expects that she/he will do almost all the UX related work, the UX specialist will become a bottleneck [8]. When a single person is responsible for a bunch of work, the work naturally is conducted serially, which will mean waiting time. Thus, we suggest involving several persons of the team in the UX work. We have studied which tasks can be performed by developers and the PO and which tasks require special professionalism in UX [10, 12, 16]. Figure 9.4 presents a summary of our findings. To conclude, POs have learned to successfully conduct workshops with users to understand the user need and to gather user feedback to improve the software [10, 12]. Developers have been able to lead the UI design work in mobile development where the platform style is prominent and the user need is well understood [11]. Moreover, developers are able to understand the instrumental quality of the software under development [16]. Instrumental quality refers to perceived utilitarian or functional quality such as usefulness in contrast to hedonic quality that provides pleasure and experiences to the user [38, 39]. Thus, we suggest integrating UX work with agile development via UX tasks instead of trying to integrate a distinct UX specialist role as such. Moreover, UX tasks should

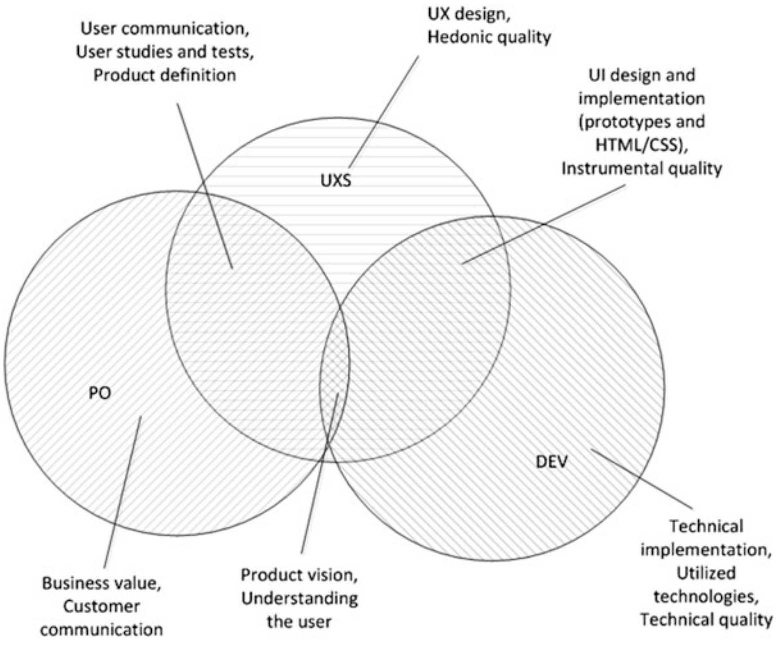

**Fig. 9.4** General task allocation between roles (Legend: *PO* product owner, *UXS* user experience specialist, *DEV* developer)

be treated similarly to other development tasks and tasks should be allocated based on professionalism and interest.

There is one fundamental difference compared to the iteration-ahead approach in the nature of UX tasks. Where UX specialists in the iteration-ahead approach seem to often produce rather ready-made UX design that the developers implement as is [6, 8, 22], the within-iteration approach emphasizes the active role of the developer in participating in UX work. This collaboration is essential for being able to design and develop during the same iteration. Without it, the UX specialist would indeed need time to produce the design ahead of development.

**Process Factors** An obvious process factor is to work within one iteration at a time; the whole team should concentrate on the same tasks at hand. Thus, the mindset should be changed from holistic UX design (big design upfront) to completing only one or a few functionalities at a time and then changing the design and related code later as needed. This does not mean that one should have no holistic idea of the system and that nothing should be thought about beforehand. It simply means that full designs are not deliberately produced, but instead, building the software can be started as soon as the team has some idea of it. To enable such a way of working, the team must allow trial and error and thus to be ready to refactor, iterate, and even discard the already built design whenever reasonable. Thus, the team should welcome technical debt considering UX design in that design decisions can be made on incomplete information. Technical debt has been defined as follows: "*not quite right code which we postpone making it right*" [40, 41]. Consequently, UX design debt can be defined as *not quite right design which we postpone making it right;* it is design that is likely to require changes and improvement later when more information is available. Allowing technical debt in UX design enables the team to work on incremental design chunks instead of having to have a complete holistic UX design available early.

Furthermore, in addition to delivering working software, the process should allow the delivery of working design. By working design, we refer to functionality that is "working enough" to be tested by users. Thus, we encourage building something that the user can actually test in the first place. In the early phase, it can be something that the user can, for instance, click through. Later it can be, for example, an added feature that has only the user interface with no or only simulated backend.

Finally, the quicker the software is on the market, the quicker the team will get actual user feedback from real use. It is difficult for a user to evaluate if they would really use some feature, if they need something, or if they would be willing to pay for a service before it is actually available [42, 43]. In addition, we have learned that users are more prone to give feedback when the system is already in use compared to when they just are asked to test something without actually benefitting from its use. Testing on actual users and developed artefacts can also save time from arranging user tests and therefore help to avoid wasteful activities.

**Tool Factors** Several factors related to tools and technologies have an impact on how effortless it is to work with the cross-functional team. UX-related tasks should be communicated via the same tool than other development tasks, for instance, via a backlog management tool [12]. Moreover, UX tasks should be chunked to pieces with similar size and level of detail compared to other development tasks [12]. Depending on utilized technologies and platforms, UX design can be designed and communicated as working software, for instance, in HTML and CSS code. If UX designers use such tools, they are able to produce the design as working software in the same time or even quicker than by drawing the design [15]. Furthermore, modern UX design tools allow varying the design fidelity [15]. Thus, the high-fidelity production design can be produced whenever feasible offering more flexibility to the timing of the work instead of fixing its delivery to a certain iteration. Finally, mechanisms and channels for collecting user feedback should be established to allow rapid feedback gathering.

**Factors Related to the Developed Artefact and the Context** The cost of building something needs to be kept in mind. The idea of the approach is to decrease the cost of unneeded and thus wasteful planning and to increase the revenue from reaching the market early. However, in some contexts, for example, when hardware is present, the cost of iteration can be higher than the cost of wasteful planning. Thus, if, for instance, a new costly piece of hardware needs to be built for each iteration, the one iteration-ahead approach might be more suitable to increase the likelihood of getting the overall system right with a smaller number of iterations. The same applies for highly regulated contexts in which heavy testing is needed before entering the markets. Thus, the within-iteration approach might be less suitable for situations where cost of error is high. However, we have no empirical data to support that planning and designing ahead of development actually would be less prone to error.

### 9.5.2 *Ways of Working in BoB Framework*

Most of the related research suggests separating product discovery and development phases [3]. We agree that there should be a high-level idea or an early vision of the product before starting to implement it. However, early product definition can be very small, and the understanding can be fostered throughout the project.

The BoB framework includes a process that consists of analysis and build tasks that are conducted continuously during the development cycle (Fig. 9.5). The process is intended for within-iteration UX work conducted in cross-functional teams. In principle, the same cycle is utilized throughout the project; thus, the process mitigates the concept of separate upfront design phases. Instead of having a particular upfront design phase, we suggest including the design and planning work into several "normal" iterations that can contain both UX design and development tasks. A UX specialist once described such an approach as follows: "*One time we started a project and we could not reach the user in the beginning. All we knew was that the user needed to be able to fill in forms. So we implemented a fillable form.*" Thus, implementation can be started with a minimal understanding of the

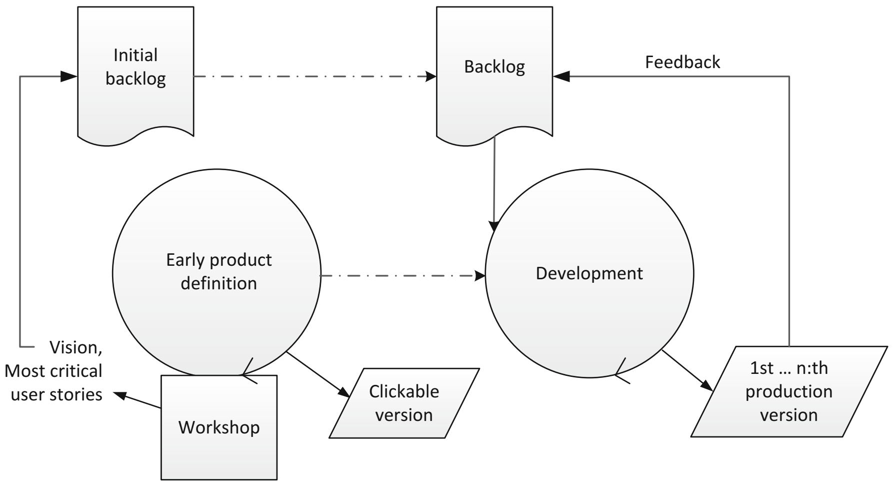

**Fig. 9.5** BoB process model for the within-iteration approach. *Dashed* lines indicate the transfer from early product definition to development

user's need and possible design. However, the early phase (design upfront or the fuzzy front end) differs from the rest of the development in that there might be nothing tangible available yet. The understanding of the project vision is minimal and communication with the user and customer might be more abstract. We suggest starting with a few short user workshops in which the product vision and most critical user stories are evolved. In between those workshops, the team works towards something tangible for the users to make it easier for all the stakeholders to understand the early product vision similarly enough. An initial backlog is formed based on the early vision and most critical user stories that have been created during the early iterations. The deliverable from this early phase can be, for instance, a partially functional prototype or a click-through template that realizes the most important user story (or stories).

After the clickable version has been evaluated with users, the team starts to work towards the first production version of the system (transfer from Early product definition cycle to Development cycle in Fig. 9.5). The UX specialist either implements the user interaction or pairs up with a front-end developer. The user interaction is built based on continuous communication within the team and together with users when needed. When a UX-related issue cannot be solved within the team, developers start to build the next task on the priority list, and the UX specialist investigates the problem until a solution is found, or until it is decided to postpone the task. The team hurries a production version to the market to start getting actual user feedback and then continues increasing the product incrementally. The team works similarly during forthcoming iterations. Thus, the approach is similar both in the early product definition phase and during the actual development phase. In addition to working software, we recommend allowing the delivery of working prototypes and partially functioning features to gain user feedback. Thus, the system can contain both fully working features and forthcoming features that are delivered for some user groups in order to allow getting early user feedback before launching the feature. This approach is especially beneficial for the majority of situations where a randomized experiment with control and treatment groups (A/B testing) is not feasible due to the size of the user population

## 9.6 Discussion

The BoB framework we introduce in this chapter offers an alternative to the commonly recommended one iteration-ahead approach. Our aim is to provide a way to focus on building the user interaction modularly feature by feature instead of running excessive user studies before the implementation. Whereas in the one iteration-ahead approach implementation is conducted on studied and tested design chunks [6], in our framework UX specialists conduct as few user inquiries as possible and small functionalities are designed and implemented possibly in hours or days after which the functionality is exposed to actual usage or usage that is close to actual (for instance, partially functional features or clickable prototypes). The functionality or some larger part of the system is then iterated based on the user feedback and new functionalities are designed based on what was learned.

The framework as such has not been in use in none of the studied companies. Although cross-functional team approach was the most preferred way to organize UX work among the participants, it was rarely utilized in the companies [8, 9, 13]. Companies preferred separate UX and development teams. However, this started to change during the research process [11], and some of the companies have utilized similar practices in their projects [12]. Those practices have included working in cross-functional teams and having UX specialists and developers working in the same iteration. In addition, Fig. 9.4 is fully based on what we have witnessed in the studied companies. We built BoB to support this within-iteration work as there has not been guidance on how to organize agile UX work in such setting.

Utilizing BoB framework requires that the UX specialist works from inside the cooperative development team. Furthermore, it benefits from a mindset that allows trial and error: when the UX design needs to be modified, the user interface will be iterated and refactored. Compared to the one iteration-ahead approach, we expect collaboration within the BoB framework to offer better visibility to the common vision, the big picture, of the project as the team including a UX specialist works in closer collaboration and makes design decisions together. We expect that the close collaboration improves the team's communication and increases developers' commitment towards UX tasks as they are more involved in UX work. Finally, we expect to get feedback from actual usage faster than in the one iteration-ahead approach as getting to the market may require only one design-development iteration instead of conducting design work first and then implementation work in the following iteration. While user studies are valuable, it is the actual usage that really validates the viability of the system.

Although we expect BoB framework to tackle several challenges in agile UX work, it does not come without its limitations. BoB works best with small, co-located teams—as agile methods do in general. Scaling up the framework is an interesting possibility for future research. Based on what we have seen in our research; such an approach has only been in use in co-located teams. Thus, as developers are in a central role in the within-iteration UX work, it might be that the approach only works when a UX specialist is co-located with the rest of the team. Therefore, it might be needed that in larger projects with several teams, there should be a UX specialist working in each team, or at least those teams that concentrate on user interaction should include a UX specialist.

## 9.7 Summary and Conclusions

In this chapter, we presented BoB framework for integrating UX work into agile development. The idea of our framework is to organize UX work in such a way that designing ahead of development during the development iterations would not be necessary. We presented both supporting factors and a process model that enables the within-iteration approach.

BoB increases possibilities to react to change, and it is to improve communication and common understanding on a team. In addition, it brings UX matters closer to developers and welcomes them to participate to the UX work as well. Thus, it is expected to mitigate the workload of the often overburdened UX specialist. The fundamental goal of BoB is to minimize the amount of upfront design and study activities and instead encourage trial and error in designing and developing for UX.

**Acknowledgment** Our research has been supported by TEKES (Finnish funding agency for technology and innovation) as part of the Cloud Software and Need for Speed research programs of DIGILE (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT and digital business).

## References

1. Cyr D, Head M, Ivanov A (2006) Design aesthetics leading to m-loyalty in mobile commerce. Inf Manag 43(8):950–963
2. Mahmood MA, Burn JM, Gemoets LA, Jacquez C (2000) Variables affecting information technology end-user satisfaction: a meta-analysis of the empirical literature. Int J Hum Comput Stud 52(4):751–771
3. Brhel M, Meth H, Maedche A, Werder C (2015) Exploring principles of user-centered agile software development: a literature review. Inf Softw Technol 61:163–181
4. da Silva T, Martin A, Maurer F, Silveira M (2011) User-centered design and Agile methods: a systematic review. In: Proceedings of the Agile methods in software development (Agile 2011)
5. Miller L (2005) Case study of customer input for a successful product. In: Proceedings of the Agile Conference '05. IEEE Computer Society, pp 225–234
6. Sy D (2007) Adapting usability investigations for Agile user-centered design. J Usability Stud 2(3):112–132
7. Roddenberry G (writer), Piller M (writer), Bole C (director) (1990) The best of both worlds: part 1 [Television series episode]. In: Berman R (executive producer), Star Trek: the next generation, USA
8. Kuusinen K, Mikkonen T, Pakarinen S (2012) Agile user experience development in a large software organization: good expertise but limited impact. In: Proceedings of the Human-Centered Software Engineering (HCSE'12). Springer, Berlin/Heidelberg, pp 94–111
9. Kuusinen K, Väänänen-Vainio-Mattila K (2012) How to make agile UX work more efficient: management and sales perspectives. In: Proceedings of the 7th Nordic Conference on Human-Computer Interaction: making sense through design (NordiCHI '12). ACM, pp 139–148
10. Kuusinen K, Mikkonen T (2013) Designing user experience for mobile apps: long-term product owner perspective. In: Proceedings of the 20th Asia-Pacific Software Engineering Conference (APSEC'13), IEEE Computer Society Order Number E5158, pp 535–540
11. Kuusinen K, Mikkonen T (2014) On designing UX for mobile enterprise apps. In: Proceedings of the software engineering and advanced applications, IEEE Computer Society, pp 221–228
12. Kuusinen K (2015) Task allocation between UX specialists and developers in agile software development projects. In: Proceedings of the Human-Computer Interaction – INTERACT 2015, LNCS 9298. Springer International Publishing, pp 27–44
13. Kuusinen K (2015) Overcoming challenges in agile user experience work: cross-case analysis of two large software organizations. In: Proceedings of the 41st Euromicro conference series on Software Engineering and Advanced Applications (SEAA'15). IEEE Computer Society (2015), doi:10.1109/SEAA.2015.38

14. Kuusinen K (2015) Continuous user experience development. In: INTERACT 2015 adjunct proceedings: 15th IFIP TC. 13 International conference on human-computer interaction 2015, vol 22. University of Bamberg Press, p 233
15. Kuusinen K (2015) Integrating UX work in agile enterprise software development. Doctoral thesis, Publication 1339, Tampere University of Technology
16. Kuusinen K, Väätäjä H, Mikkonen T, Väänänen K (2016) Towards understanding how agile teams predict user experience. In: Integrating user-centred design in agile development. Springer, Cham
17. Cockburn A, Highsmith J (2001) Agile software development: the people factor. IEEE Comput 34(11):131–133
18. Beck K et al (2001) Agile alliance. Principles behind the Agile Manifesto. Available at: http://agilemanifesto.org/principles.html
19. Highsmith J, Cockburn A (2001) Agile software development: the business of innovation. Computer 34(9):120–127
20. Schwaber K (2004) Agile project management with Scrum, 1st edn, Microsoft professional. Microsoft Press, Redmond
21. Salah D, Paige R, Cairns P (2014) A systematic literature review on agile development processes and user centred design integration. In: Proceedings of the 18th international conference on Evaluation and Assessment in Software Engineering (EASE'14). ACM, Article 5, 10 p
22. Ferreira J, Sharp H, Robinson H (2010) Values and assumptions shaping agile development and user experience design in practice. In: Proceedings of the XP 2010, LNBIP 48:178–183
23. Gulliksen J (1999) Bringing the social perspective: user centred design. In: HCI (1) 1999, pp 1327–1331
24. Cagan M (2012) Dual-Track Scrum. Blog post 17 September 2012. Available at: http://www.svproduct.com/dual-track-scrum/. Accessed 9 Mar 2016
25. Zaman K (2014) Dual track Scrum. Scrum alliance member article 18 December 2014. Available at: https://www.scrumalliance.org/community/articles/2014/december/dual-track-scrum. Accessed 9 Mar 2016
26. Patton J (2014). User story mapping: discover the whole story, build the right product. O'Reilly Media, 324 pp
27. Ferreira J, Sharp H, Robinson H (2011) User experience design and agile development: managing cooperation through articulation work. Softw Pract Exp 41(9):963–974, (Wiley)
28. Isomursu M, Sirotkin A, Voltti P, Halonen M (2012) User experience design goes agile in lean transformation—a case study. In: Proceedings of the Agile Conference (AGILE 2012), pp 1–10.
29. Hodgetts P (2005) Experiences integrating sophisticated UX design into agile process. In: Proceedings of the Agile Conference 2005, IEEE, Denver, CO, pp 235–242
30. Lee JC (2006) Embracing agile development of usable software systems. In: Proceedings of the Conference on Human Factors in Computing Systems CHI 2006. ACM, pp 1767–1770
31. Eisenhardt KM (1989) Building theories from case study research. Acad Manag Rev 14(4):532–550
32. Eisenhardt KM, Graebner ME (2007) Theory building from cases: opportunities and challenges. Acad Manag J 50(1):25–32
33. Yin RK (2003) Case study research: design and methods, 3rd edn. Sage, Thousand Oaks, 181p
34. Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14:131–164
35. Gulliksen J, Göransson B, Boivie I, Blomkvist S, Persson J, Cajander Å (2003) Key principles for user-centred systems design. BIT 22(6):397–409
36. Chow T, Cao DB (2008) A survey study of critical success factors in agile software projects. J Syst Softw 81(6):961–971
37. Gulliksen J, Göransson B, Lif M (2001) A user-centered approach to object-oriented user interface design. In: van Harmelen (21), chapter 8

38. Holbrook MB, Hirschman EC (1982) The experiential aspects of consumption: consumer fantasies, feelings, and fun. J Consum Res 9:132–140
39. Lantos GP (2015) Consumer behavior in action: real-life applications for marketing managers. Routledge, New York
40. Kruchten P, Nord RL, Ozkaya I (2012) Technical debt: from metaphor to theory and practice. IEEE Softw 6:18–21
41. Cunningham W (1992) The WyCash portfolio management system. OOPSLA' 92 Experience report
42. Anastassova M, Mégard C, Burkhardt JM (2007) Prototype evaluation and user-needs analysis in the early design of emerging technologies. In: Human-computer interaction, Interaction design and usability. Springer, Berlin/Heidelberg, pp 383–392
43. Van Kleef E, van Trijp HC, Luning P (2005) Consumer research in the early stages of new product development: a critical review of methods and techniques. Food Qual Prefer 16(3):181–201