Chapter 2
User Integration in Agile Software Development Processes: Practices and Challenges in Small and Medium Sized Enterprises

Oliver Stickel, Corinna Ogonowski, Timo Jakobi, Gunnar Stevens, Volkmar Pipek, and Volker Wulf

Abstract HCI and CSCW research as well as practice has strongly indicated the value of integrating (end) users in software development processes. Such integration can help address actual needs and wants, to avoid undesirable developments and to strengthen the User Experience of a product. A user-focused approach to software development has some conceptual overlap with agile software development practices, such as quick and iterative (user) testing. However, out in the wild, organisations seem to have difficulties actually mapping user-centered development with agile processes for a variety of reasons ranging from organisational or hierarchical aspects up to financial issues. This problem seems specially prevalent in Small and Medium sized Enterprises (SMEs) where such constraints can be even tighter than in larger organisations. To help understand those problems and to identify possible solutions, we turned to three quite different German software SMEs, varying in size, market focus and organisational structure. By way of qualitative field studies, we were able to identify key roles and tools as well as methodological, organisational and analytical practices and challenges in integrating (end) users into agile software development.

Keywords Agile software development • User centered design • User feedback • Case study • Qualitative study

O. Stickel (✉) • C. Ogonowski • T. Jakobi • V. Pipek • V. Wulf
University of Siegen, Siegen, Germany
e-mail: oliver.stickel@uni-siegen.de; corinna.ogonowski@uni-siegen.de; timo.jakobi@uni-siegen.de; volkmar.pipek@uni-siegen.de; volker.wulf@uni-siegen.de

G. Stevens
Bonn Rhein-Sieg University of Applied Sciences, Augustin, Germany
e-mail: gunnar.stevens@h-brs.de

2.1 Introduction

Software has become an invaluable part of private and professional life all over the world. This has led to Usability and User Experience[1] becoming increasingly important factors for the success or failure of ICT systems. While this obviously holds true for all sort of systems allowing user interaction, for the purpose of this contribution, we will focus on software systems. For the software world, we have solid research [13] as well as norms such as the DIN EN ISO 9241 suggesting that integration of (end) users in all phases of a development project is one of the most central factors for positive UUX.

Looking at the economically important sector of Small and Medium Sized Enterprises (SMEs) however, we frequently find deficits in the incorporation of UUX methods into software development. Hering et al. [14] indicate factors such as financial, logistical, hierarchical or methodological issues that hold the SME sector back with regards to the systematic integration of users and user feedback in development processes. Furthermore, norms and process models such as the aforementioned ISO 9241 or user-centered design (UCD) often lack clarity regarding the *actual* implementation of user integration and how to fit this into established process models in organisations.

Our contribution addresses this research/practice gap by helping to identify relevant issues for SMEs when dealing with user integration and presenting solutions as well as best practices evolved in these organisations. We grounded our work in a practice-based, socio-technical understanding of Human Computer Interaction (HCI) and Computer Supported Collaborative Work (CSCW) [33]. Consistent with this base, we chose qualitative case studies in three contrasting German software SMEs as our main research instrument:

Foo[2] is one of the largest German SMEs in the software business focusing on end users with quite nuanced processes for the integration of user-centered methods and agile development processes.

Bar is a relatively large SME (if decidedly smaller than Foo) producing software and hardware for end users. Bar's focus on UUX has a briefer history and smaller extent than Foo's.

Qux is a very small, design-driven software company which mainly fulfills orders, i.e. with no direct end user market.

Based on our fieldwork in all three organisations, we were able to identify *Roles*, *Channels and Media* as well as *Interpretation and Filtering* of user feedback as the three main themes moderating (and moderated by) the success or failure of user integration in agile development processes. In the following sections, we first give

[1]From here on, we will abbreviate "Usability and User Experience" as UUX. For the purpose of this chapter, we do not need the distinction between more task-focused and more ludic aspects.

[2]All organisation names as well as all personal names in this contribution are anonymised for privacy reasons.

an overview of the relevant state of the art before reporting our results and discussing them with a focus on these three themes.

2.2 Related Work

In this section, we present an overview of the relevant scientific background and literature, starting with a very brief primer on agile software development, leading up to the relevance of user integration for positive UUX and finally the synthesis of both aspects.

2.2.1 Agile Software Development

Agile software development [2] refers to relatively new paradigms for structuring ICT projects such as Scrum [25] or Kanban [1]. Agile methods differ from classical process models such as the “waterfall” in rejecting the notion of a “heavy”, largely predefined and pre-planned project which is then processed step by step. Instead, agile methods take into account changes occuring during software projects and propose to prepare for and embrace them [31]. This results in four central values as codified in the *Agile Manifesto* [2]: (1) Individuals and interactions over processes and tools (2) Working software over comprehensive documentation (3) Customer collaboration over contract negotiation (4) Responding to change over following a plan.

2.2.2 User Integration for a Better UUX

Existing literature provides several reasons for integrating users and customers into the design process including improved UUX as well as political, economical and ethical considerations [34]. For example, Participatory Design (PD), arguably the earliest systematical approach for active user involvement in software development, originates in workplace democracy movements [4, 9], supported by trade unions. However, commercial software companies also discovered and implemented PD, valuing methods such as collaborative storyboarding or group elicitation approaches [11]. *Active* user participation was deemed to be effective since actual users of a product were understood to know their own perspectives and needs best [22]. While most “original” – political – PD approaches do not necessarily consider positive UUX as a core focus, various understandings of PD have evolved with different accentuations. For example, the American school of PD dropped the political framework and rather pursued the development of more efficient products [15], thus taking user integration in a more UUX-focused direction.

Later approaches to user integration in ICT development projects include Integrated Organisation and Technology Development (OTD) [32] as well as STEPS

[12]. Both approaches are normative software development models that involve close collaboration of users and developers. Both are also more focused on their application in organisations. However, as they are designed for a very close and rather intricate collaboration of developers and customers, they are not easily usable for the development of mass-market off-the shelf software – especially not for SMEs, considering their often limited resources [13]. Looking at the earlier stages of ICT projects, we should also mention von Hippel [30], who has long focused on User Driven Innovation and its benefits. However, the focus on early phases also limits this approach.

Current trends include Design Case Studies which involve significant user integration [33] as well as Infrastructuring as a more holistic perspective on how to understand the development of socio-technical systems [23]. Managing user integration in ICT projects also increasingly relates to user-centered design (UCD). UCD, as codified in ISO 9241-210 can be seen as a normative design and development model that argues for user integration in all phases of a development process. Consequently, the UCD ideology specifically views the user as an asset of the product development process. Furthermore, unlike older models, it explicitly focuses on generating a positive UUX as well. Within a UCD process, users should be included in early phases (ideation) and user research should be conducted. This covers everything that helps to understand who the users are, what their system of values and requirements are and so on. Further on in the process, users can participate in mock-up generation or evaluation and similar activities before finally being consulted in the evaluation of releases. Since UCD was developed to be adaptable to existing software engineering approaches, its specification leaves room for tailoring to a local context which in turn needs interpretation and negotiation in the form of interaction between users, designers, engineers and other stakeholders.

2.2.3 Synthesis and Research Gap

Agile methods favour 'customer'-focus [2] while UCD and UUX obviously address 'users'. Especially in corporate settings, the customer does not usually coincide with the end user. However, both ways of thinking highlight stakeholders and their needs instead of favouring a process-focused view. In this regard both approaches share quite relevant characteristics [6]. It is, however, much less clear how to integrate them on a practical than a conceptual level:

There are multiple positive reports on adaption and integration attempts of UCD and UUX. Isomursu [16], for example, presents a single case study on a multinational corporation and its shift to agile methods. Also, Sy [29] reports on beneficial effects on a product's UUX in the case of a big corporation through the combination of two measures: firstly, the development process was restructured in favour of a more agile procedure, and secondly, the reporting of usability testing activities were modified to match the agile cycles. However, it has also been noted quite frequently [10, 19, 21] that the actual implementation of UCD – and more generally, the optimal combination and positioning of different UUX methods – is not yet well understood in practice. Hence, there have been different scientific workshops and tutorials, e.g. [18] as well as suggestions for procedural models or frameworks to facilitate integration. Silva et al. [26], for example, base their framework on an Interaction Design Lifecycle and specifically include design cycles into the agile process. Beyer [3] focuses on UUX professionals and how to integrate them in agile environments, not least by facilitating understanding for UUX development strategies. Scrum roles themselves are also regarded as relevant for the successful integration of UUX and Agile. Singh [27] identifies the Product Owner (PO) as the most crucial role for such attempts and states that POs are often overwhelmed since they have to coordinate many stakeholders, artifacts and ceremonies and are not necessarily qualified in UUX. This leads the authors to propose the appointment of two POs, one of which focuses on more traditional responsibilities in the Scrum model while the other one's responsibilities lean towards UUX [27].

Overall, literature suggests manifold thematic relations between UCD/UUX methods and agile software development. There have also been investigations and practical attempts to integrate both approaches, leading to some beneficial results in practice as well as some more theoretical concepts. However, prior work strongly indicates that the understanding of UCD/UUX and agile still leaves many gaps, especially regarding the systematic understanding of the actual practices and challenges faced by organisations *in the wild* [8, 10]. This contribution is an attempt at helping to fill this gap by way of three comparative case studies with a focus on the domain of SMEs.

2.3 Cases

In the following sections, we will first describe our three cases in more detail before summarizing them in a tabular form.

2.3.1 Foo: A Very Large SME with Established UUX Practices

Foo is a large SME with about 500 employees and a strong corporate focus towards UUX and user integration. Actively pushed by the company's management, this culture has evolved over many years. During those years, Foo has experimented with different approaches towards development and/or user integration, ranging from traditional waterfall models to Participatory Design projects (explicitly framed as such). Foo's product portfolio is centered on software systems for end users with an emphasis on personal and organizational finance administration and management tools. We mainly worked with one project team within Foo which is responsible for iFin, a tool for personal finance management. iFin is a mass-market product, cross-platform (mobile and desktop, multiple operating systems) and is developed in an agile fashion, utilizing Scrum. iFin has settled on 4-week sprint cycles and the agile team consists of developers and designers. Some other roles we will be referring to later are not part of the Scrum team – they are asked to work with the Scrum team as needed. Hence those roles do not have to work in fixed sprint lengths but – given the need to collaborate with the 'core' Scrum team – they are aware of the sprints and their work is moderated by those cycles and other agile practices developed by the Scrum team.

2.3.2 Bar: A Big SME with Emerging UUX Practices

Bar is a large SME with about 200 employees. Until recently, Bar focused on customer home electronic components. Especially in the area of home network components, Bar has developed nuanced processes and competences. However, more recently, the company decided to develop a line of Smart Home components which were about to be launched on the market at the time of our study. This led Bar to focus more strongly on software development in general and interface design in particular. Due to the increased amount of user interaction with smart home devices in comparison to more passive network electronics, UUX was explicitly addressed, too. Originally a hardware-developing and engineering company, Bar is used to managing projects with a strongly phase-oriented process model. However, at micro level, at least the software team reportedly self-organizes using Scrum. Studying the case of iHome development, we found several more ways in which Bar departed from the phase-oriented path and tended towards more agile methods. The complexity of a Smart Home system, the multitude of (also external) parties involved and the stronger emphasis on interaction components led to a mixture of milestones and agile ways of completing them. Integrating end users into the evaluation of iHome prior to market launch was deemed especially important by Bar. Our work with Bar focused on the smart home team, their emerging agile development processes as well as their in-house user test sample for working with and evaluating prototypes, both in terms of functionality and UUX.

2.3.3 Qux: A Small, Design-Driven Software Company

Qux is a growing but still quite small software developing company of just 11 employees. They offer development and consultancy services and design of innovative software and mobile apps as well as digital products in areas such as the Internet of Things, smart home, energy and e-mobility. Being a service company, Qux's focus is less on selling to end users directly but rather on projects for their customers, who provide the products and services to end users. The company has successfully established a flat team hierarchy. It is only divided in two units: design and development, which are supplemented by the functions of both CEOs (Scrum Master/Project Manager and Creative Director) and Social Media Marketing. One of the CEOs, who is responsible for project management, also manages the commercial tasks of the company and hence does not carry the title of Product Owner. Related to the company's hierarchy, Qux has a corporate culture and image with a strong focus on communication and exchange between all employees; design-driven development; UUX and decision making. Their agility is reflected in the management of their projects. Projects were conducted using 2- to 4-week sprints, depending on the project's size. Customers play an active role in the design and development process. In regular sprint reviews they have to provide additional feedback about the design process and results with respect to current developments of the market or internal strategical decisions in order to keep project progression flexible and close to market trends. Transparency and continuous communication with customers is a key issue for Qux too.

2.3.4 Comparative Overview of Three Cases

For a comparative overview of the three cases and their characteristics. Table 2.1 provides details and summarizes data with respect to size, study focus, agility and peculiarities of the SMEs.

2.4 Method

In this section, we will explain our methodology and our analytical process. Subsequently, we will provide an overview of our data and coding scheme.

2.4.1 Study Design and Data Collection

How does [Foo | Bar | Qux] integrate user input and feedback into their agile software development process and how does this relate to UCD?

This was the basic research question motivating our study. It is important to note that we did not approach the field with a focus on up-front theory but rather based our approach on open, field-driven research, inspired by Grounded Theory (GT) [28]. Therefore in each case, we went quickly into the field where we iteratively developed our understanding of the company's practices as well as our research strategy according to our findings. We deemed a (field-)data-driven approach to be important given the disparities between theory and practice and the ambiguities described in the state of the art.

Table 2.1 Case summaries

	Foo	Bar	Qux
Size	About 500 employees	About 200 employees	11 employees
Product portfolio	Wide variety of software products for mobile and desktop, focused on finance administration on the personal level as well as for the SME and nonprofit sectors	Variety of products concerned with home networks. More recently soft- and hardware for the smart home including heavy coordination with third parties	Software solutions focusing on mobile applications, front and back end applications for the domains of energy, smart home, renewable energies, e-mobility and the Internet of things
Study focus	iFin, a cross-platform personal finance management tool	iHome, a soft- and hardware ecosystem for smart homes	No specific project, lateral study through the company
Agility	Scrum team utilizing 4- week sprints. Core Scrum team (mostly developers) is supported by other teams such as e.g. an inhouse usability lab who do not work in formal sprints – however, the Scrum team sets the overall pace	Complexity led away from sprints to milestone- oriented development. Requirements engineering upfront, but highly iterative within three main phases: proof of tech-concept in the wild, proof of combination of HW and UI concept, beta testing	Scrum-oriented project management with 2- to 4-week sprints based on project size. Company's philosophy follows principles of user-centered design. Active integration of customers in design and development by regular sprint reviews
Peculiarities	Long company history and company culture of user-centricity (established over years). Three product owners instead of one for iFin	Established their own in-house testbed, semiprofessionalised. Project manager with sole direct contact to users, defines usability concepts	Mainly business to business but target is often mass-market. Project manager is not framed as project owner. High transparency to customers

In total, we conducted 15 interviews. Seven of them were at Foo and four at Bar and Qux respectively. Within the iFin team at Foo, we conducted interviews with a Product Owner (PO), Social Media Management (a one-person team), the head of the support team[3] for all products (not just iFin), a member of the support team specialised in iFin and a developer as well as two members of the in-house usability lab. We also conducted participant observations during usability tests in the in-house lab (3h) as well as during a Scrum planning meeting (4h). At Bar, we conducted interviews with the heads of the development team, the Design and Verification and Testing (DVT) team, the product marketing team as well as the responsible PO. At

[3]To be clear: Foo's support team is the user support department, i.e. the staff responsible for helping customers with issues. The name 'support team' is actually an in-vivo code from the fieldwork at Foo.

Table 2.2 Data index

ID	Description	ID	Description
I-F-01	Product owner	I-B-01	Product manager
I-F-02	Social media officer	I-B-02	Head of development
I-F-03	Chief of support	I-B-03	Head of design, verification and testing
I-F-04	Customer lab	I-B-04	Head of marketing
I-F-05	First level support	I-Q-01	CEO 01: scrum master & head of project management
I-F-06	Software developer	I-Q-02	CEO 02: creative director
I-F-07	PO other project	I-Q-03	Senior art director UI/UX
O-F-01	Scrum sprint planning meeting	I-Q-04	Mobile developer
O-F-02	Two usability tests		

Qux, we conducted interviews with two of the three CEOs who also acted as Scrum Master/Head of Project Management (PM) and Creative Director (CD) respectively. Similar interviews were conducted with a Senior Art Director UI/UX and a Mobile Developer.

In Table 2.2, we have indexed all interviews and observations. For clarity: I-F-04 was an interview with two participants (the full staff of Foo's in-house usability lab); I-F-07 was an interview with a PO for a different product team than iFin since this PO was referred to us as one of the central experts in regards to agile software development and UCD in the company[4]; and in cases such as I-B-03 and I-Q-01, one person fills multiple roles.

The interviews lasted between 60 and 120 min and were recorded as well as transcribed pragmatically, i.e. full verbatim transcriptions utilising only markers for salient events such as laughter, peculiar facial expressions or breaks. However, we did not include micro-expressions, precise break times, detailed pitch analyses, etc. since we did not deem such data necessary for our research interest in practices and challenges. All interviews utilised a guideline which evolved in the field, led by the field. The interview language was German in all cases, the quotations in this contribution are translated. Transcripts were supplemented by handwritten field notes and memos (about 25 pages). Furthermore, we gathered artifacts such as user stories, bug reports or usability reports, mainly at Foo because of its bigger size and the availability of many such artifacts as well as the fact that Foo was our first case study and the data helped us to open up the field. Finally, we supplemented our interviews by multiple further inquiries to the interview partners via Skype, phone and email during the analytical process whenever relevant questions arose. Brief descriptions of the data sources can be found in Sects. 2.3.1, 2.3.2, 2.3.3 and an index in Table 2.2.

[4]At this point in the analytical process, it had already become clear that the intersection of those two topics would be central to our study.

2.4.2 Data Analysis

All data and artifacts were subsequently coded axially and selectively using a GT approach [28]. However, we do not claim to have established a ‘Theory of UCD and Agile’ – we feel that such an encompassing theory would necessitate multinational and even more contrasting cases as well as a longer period of time. Rather, our analytical process followed GT methodology and can serve as one of many pieces in a more comprehensive puzzle towards a theory. To clarify even more: we oriented ourselves on Thematic Analysis [5] which, essentially, is GT without the overhead of extensive theory building but with the option to add that on top iteratively. The coding process started immediately after the first interview and was continued and evolved throughout the research activities. During the field research phases, we held weekly discussion and mirroring meetings regarding the coding activities in our research group. This also included researchers who were not active in the field, some not even in our research project at all. These researchers helped by asking questions those working in the field did not think of, forcing the latter to explain a significant amount of tacit information. The coding structure continued to change up to the point when the gathered data no longer added significant new insights (saturation). This also helps to explain the different contents of data collected in the three cases – with the evolution of a denser coding scheme, (transferable) insights led to saturation points more quickly as per the intention of GT-inspired approaches.

Table 2.3 provides an overview of the theme structure as well as examples of sub-codes for each case. The three central themes boil down to “Roles”, “Channels and Media” and “Filtering and Interpretation”. Our report on results which follows in the next section is also oriented on this structure.

2.5 Results

In this section, we will report on the three most important themes as listed in Table 2.3 as well as their interrelations, starting with the *Roles*, leading up to *Channels and Tools* and finally, aspects of *Filtering and Interpretation*.

2.5.1 Roles

2.5.1.1 Foo

With Foo, we found multiple roles to be in contact with users. The support team obviously has most points of contact since they are confronted with a wide variety of user issues on a daily basis. However, they are not only trained to solve those issues but also to try and understand where they come from and ask for more feedback than strictly necessary to solve the issue in order to provide input for

Table 2.3 Examples of analysed sub-codes

Themes	Foo	Bar	Qux
Roles	PO, multiple POs, Social media management, support team, developer, role empowerment, differentiation of POs' skills, in-house usability lab	PO, DVT, development, user contact, categorising feedback, managing feedback, market analysis, in charge of usability, functional tester, coordinator	Project manager, creative director, senior art director, social media manager, developer, quality management, testing by noninvolved employees
Channels & tools	Email, chats, phone, letter, forums, facebook, twitter, TFS, bug tracker, daily stand-up, Scrum, sprint, user story, facebook, daily stand-up, sprints, open and honest communication, channels towards the user, channels from the user, app stores, blogs, grapevine, coffee corner	Bug-tracking, office grapevine, technical proof of concept, PO as field tester, friendly user testing, product specification tool, employee participation, missing standard tools, wireframing, outsourcing user studies, gap user value – integration, iterative development, milestones, chat, phone, virtual seminars	Bug-tracking, test cases, user story, daily stand-up, coffee corner, friendly user testing, app stores, market research customer sprint review, third-party services, email, phone facebook, missing standard tools, (non) filtered feedback, integration in management software, taking a step back as designer
Filtering & interpretation	XYZ (long-planned feature preventing certain feature requests from being implemented), what does the customer actually want? Company culture, grapevine, discussions, database, lead users, conflicts, mood	Prioritisation via frequency, ticket-system, expenditure processing feedback to make it useful, log files, text-data, pictures, video-data, limits of outsourcing feedback, sharing feedback with third parties in the project	Frequency, missing metadata, feedback requests, sorting & editing, communication with customers, ticket-system, testers' aptitude, reliability in third party services, lab tests as stress situation, environment control

product development. To this end, Foo has kept their support team in-house, located near the development, management and other teams. They are also actively trying to foster a culture of deep and long-term engagement with, as well as knowledge about, Foo's products. To let a first level support employee speak for himself:

> I've been working for Foo for about ten years now. I can use the software blindfolded. I can find problems and difficulties while standing on my head. I-F-05

Foo also has an in-house 'customer lab', which is a traditional usability lab, staffed by two UUX-experts. They carry out structured user testing at the request of the iFin team and report to the Product Owner. However, the customer lab is not part of any team as such – in essence, they offer a service to all of Foo's development teams. Furthermore, Foo also has a defined role for Social Media management (SMM). The SMM tries to engage with users by way of providing them with information, monitoring discussions, trying to mediate if necessary and very consciously tries to get a "feeling for the mood" (I-F-02) on Social Media in regards to Foo's own products as well as the competition. Like the support team, the SMM is not part of the Scrum team as such and also has some other duties in the company (such as maintaining blogs not connected to iFin). However, the SMM's main focus is iFin and by far the majority of her work time is spent on this project.

As in established Scrum doctrine, we found the role of the PO to be the central hub within the different approaches of user integration and user contact in Foo. There are two notable observations in regards to Foo's PO structure for iFin: First, there are actually three POs. One manages daily affairs such as codifying user stories; the second one focuses on the epics and the third has a background in design. The PO-team's skills compliment each other; however, they also consult with internal experts (such as the SMM) on a case-by-case basis. Second, while it is certainly in the Scrum-spirit for the PO to represent end-users (and hence, to engage with them as well), some of iFin's long-term users even have the PO's phone numbers and call them occasionally, especially when something in the product changes in a way they do not like. Software developers themselves do not usually have user contacts in Foo.

2.5.1.2 Bar

Bar has a product-oriented organisational structure based on Business Units. The company's software development teams are familiar with Scrum methods and use sprints when working with their core product line of home networking systems. However, the general process of developing a product is not iterative:

> The usual procedure here is the so-called phase model. We divide this into five parts: Evaluation, conception, planning, and prototyping [and finalisation] phase - that's where you can see we have a background in hardware. [..] Within the development division we principally organise ourselves using Scrum. Not textbook-style, but tailored a little to Bar's needs. We do daily stand-ups, though, and plan our sprints with items. (I-B-02)

The specific requirements and the heavy software focus of iHome, however, have influenced the general project management and development process of Bar towards a more agile and iterative development, which is reflected by the emerging roles, tools used and integration of user feedback. The latter is generally rarely surveyed or integrated into Bar's development cycles, as user interaction with its products, especially via software, is only an optional feature. Therefore, with the decision to develop iHome, Bar enters new territory. For a better understanding of existing feedback practices, it is worth noting that our interviews focused on the development process of iHome before market launch. At this point, Bar had no experience in handling customer feedback after rollout but had only begun to specify strategies.

Direct user contact with a friendly user group is limited and structured in clear channels through a fixed sample of users as well as internal testing (more on this below). During development, Bar's 'Design, Verification and Testing' division was the central role responsible for testing and validating new software releases with regards to bugs and completeness compared to the requirements:

> Since the DVT is our last line of defense, they have to check somehow what has been developed . . . Meaning they always compare the requirements with the result [result = a release] (I-B-01)

At the time of writing, Bar has also established a support team structure intended to work closely with users with the explicit goal of feeding back to product development. Like Foo, Bar also utilises Social Media as well – however, with Bar, Social Media work is co-located within the marketing division, whereas Foo has a separate, explicit organisational role for Social Media management. Bar also planned to hold web based seminars to explain possible usage scenarios to customers and has included a direct chat feedback mechanism in their software. Furthermore, the software also features classic support options via mail and phone. Bar's development team is more distributed than Foo's, including more external partners, with the in-house development team focusing on coordination and conception. As with Foo, Bar's central role for UUX is the business unit's PO. He handles all reports and user feedback and makes all decisions in regards to UUX. During the user testing phase, he mainly drew feedback from the office grapevine and the bug tracking system used by test households:

> [. . .] Then, they [user feedback and feature requests, consolidated by the DVT] came to me. [. . .] and I had to go back to the wireframe or make clear how this and that is intended [. . .]. (I-B-01)

The PO consults with external companies on a case-by-case basis. He has direct user contact, mainly for concrete, deeper enquiries and user problems within the test sample. This approach, however, is rather unstructured and is either prompted by a specific problem description via the bug tracking system or by friendly users directly approaching the PO. In addition to the PO with his quite direct channel, DVT sometimes has contact with friendly users, but less frequently and only when clarification is needed on a bug reported. While these structures have proven successful for the beta test, up to this point, Bar has not yet decided which department should take responsibility for handling feedback from real customers after the launch, nor how to manage underspecified feedback.

2.5.1.3 Qux

Qux has a very flat hierarchy which splits up into a Design- & Development unit, Social Media Marketing as well as the roles of both CEOs who act as Scrum Master & Head of Project Management (PM) and Creative Director (CD), respectively. The PM is responsible for internal quality control of concepts and releases while the CD and the Senior Art Director UI/UX (AD), located within the Design & Development unit, manages all UUX aspects. Hence Qux has formed a structure where the PM takes on what might be called the more managerial aspects and the AD the user-focused ones, in comparison to Foo and Bar who subsume both aspects under the role of their respective POs.

In contrast to the typical role of a PO, in Qux the PM takes the responsibility not only for these tasks but also for additional functions of the company. Besides the project management of all projects realised by the company, he manages commercial tasks which is why they framed his role as PM. Based on the number of employees, there is still no need for several Product Owners, who are responsible for single projects or markets. This might change if the company grows further.

A central distinction of Qux as a software company in comparison to Foo and Bar is that their customers are generally not their end users. Hence, we find roles such as support teams and product-specific social media engagement not within Qux but rather within their portfolio of customer organisations. Wider user tests (and hence roles with user contact) are also outsourced to third parties or the respective customer organisation takes charge of those activities itself. Qux also actively asks the customer organisations for feedback after each sprint.

Furthermore, this structure brings with it a certain fluidity of roles. On a case-by-case basis, Qux leverages all of its staff as well as friends and family for ad-hoc testing and feedback. This culture is illustrated quite well by the actual Senior Art Director UI/UX:

> [...] my father, who has no affinity for such things [ICT]... I really like to just hand him stuff [beta versions] – just to see what he does. (I-Q-03).

The dynamic feedback loops between roles and units all converge on the PM. This approach of internal testing is based on a quite explicit corporate culture focused on user-centered design which encourages everybody working on a project to constantly take a step back and actively try to view the product through the eyes of a customer as well as a user:

> I think we are quite good in putting ourselves into those roles [users] [...] When somebody is working on a project, we also try to put him together with a colleague working on a different project [...], to get a different view. I think that's really important. (I-Q-02)

2.5.2 Channels and Tools

2.5.2.1 Foo

Central to Foo's agile Process is Microsoft's TFS which is used as a code repository as well as for handling and prioritising the backlog and supplementary data such as technical logs and feature requests as well as usability test reports. The developer especially, in addition to the POs, utilise TFS to manage and track iFin's development.

The support team utilises email, phone, fax, letters and chat as well as product-specific web-forums to engage with users directly, although the forums are focused on a *"customers help customers"* (I-F-03) approach. User feedback is taken from the support-specific ticket system and is put into the TFS if deemed valuable (more on this distinction in Sect. 2.5.3).

The Social Media manager mainly utilises Facebook and Twitter and, to a lesser extent, Blogs as channels to interact with and include users. Notably, she does not use any special Social Media management tool. She tries to contextualise user feedback as much as possible by utilising the rich data provided by Social Media. Subsequently, she directly engages with the POs in about feedback via email or face-to-face conversations. It is notable that she does not use the TFS even though she has access to it. Furthermore, regular surveys utilising the Net Promoter Score [17, 24] are carried out. When problems occur such as server outages, known bugs or similar issues, the Social media management informs customers via available channels and, more importantly, keeps them up to date. An example from I-F-02 was a bug occurring after an update which crashed the app immediately after starting it. A bug-fix was implemented and submitted to the app store very quickly but due to the approval process in the store concerned, the update needed time to be made available to the customers. The SMO kept the customers informed every step of the way which received positive feedback.

The customer lab's main channels and tools are traditional user tests with Thinking Aloud and sometimes Heuristic Evaluations and Cognitive Walkthroughs, although they also use methods such as Contextual Inquiry-inspired approaches, even in users' homes. The CL usually utilises series of tests with 5–20 participants and frames the results as comprehensive reports in a structured format. These are subsequently put into the TFS for the POs. Notably, it is also possible for everybody in the development team to tune into live video feeds from the usability testing sessions, although it has been expressed in I-F-04 and I-F-06 that developers do not usually do this, stating that the *"reports are enough"* (I-F-06). Tests in the CL are only carried out by request of the POs, the management or other decision making roles.

Apart from the TFS, the POs also have product-specific email accounts for free-form feedback which can be reached by the users from within iFin. Furthermore, the POs actively monitor as many app stores and similar places on the Web where users leave feedback of some sort. Foo even developed an in-house tool, specifically for the purpose of aggregating such reviews and making them manageable. As mentioned before, lead users sometimes contact the POs in person, utilising phones as well as email. Foo's POs also receive a certain amount of automated use tracking data. However, this is reserved for very specific and heavily debated cases due to privacy concerns. In line with Scrum practice, one of the most central tools for Foo's POs are User Stories which are built, maintained and utilised without any company- or project-specific peculiarities.

Central channels and media in Foo also include sprint review and planning meetings, meetings and discussions among the POs as well as daily stand-ups. As already indicated, only the Scrum team itself is included in those activities by default. Other roles such as the customer lab or the SMM can and will be asked to join specific meetings on request. However, a significant amount of coordination, discussion and other (meta-)work is also done without any formal media or channel: *All* interviewees in Foo talked about the importance of *"Flurfunk"* (I-F-01) (literally "corridor radio"– the office grapevine), coffee corners and informal meetings for coordination, sharing and discussing user feedback and user perspectives.

2.5.2.2 Bar

Within the unit we worked with, software development was initially structured in sprints, whereas the overall project plan featured three main phases: technological proof of concept, bringing UI and hardware together and, finally, beta testing with friendly households and bugfixing. Due to the complexity of both the system and the project itself, Bar switched to a rather milestone-driven development cycle, in which iteration was promoted. In particular, Bar decided to test its system in the wild during its hardware development, rather than relying on lab testing only. To our understanding, this was already a major difference compared to the usual development processes, which can be perceived as one example of acknowledging the need to involve users in earlier stages of development. Regarding the product specification, Bar utilises a custom in-house database system geared towards product management in which all requirements and properties of the product are held and maintained. Generally, the main features and style of the product were defined up front in this database which is used for all Bar products. Wireframes of the final system were developed quite early in the development process of iHome and can be understood to be similar to traditional target specifications for internal purposes as well as coordinating artifacts with external contractors. While usually static for Bar's other products, it turned out that the requirements specification of iHome called for much more flexible handling compared to typical products in the system, where there are fewer user interfaces. Central documents like wireframes were therefore included in the specification system, but were frequently updated throughout the whole project.

> [...] Meaning they [the DVT] always compare the requirements with the result [a release] and can refer to the wireframe [...] [interviewer asks how the wireframes changed during the development process] Well, they stayed relatively stable in scope [...] here and there, there were adaptions [...] (I-B-01)

While testing the technical proof of concept was limited to members of the software development division, systematic user testing regarding the UI is only applied when all desired features, as specified by the wireframes, have already been implemented. The focus in this phase is not on finding innovative new features or investigating end-user appropriation but rather to purposefully shape UI-components and interaction flows. To this end, Bar has joined forces with an external partner in order to establish a Living Lab [20] infrastructure as a test bed: About 30 households are given the product a few months before rollout in order to test it in their homes. These tests take place without instruction or rules apart from a commitment to actively use the system and test specific features after updates. Bugzilla has been implemented as a channel where users can input tickets. This is supplemented by occasional informal exchanges face-to-face. During the Living Lab phase, Bar also recognised that users did not always seem to log all their problems into Bugzilla, especially when the problems in question did not relate to hard and evident bugs:

> [. . .] If somebody had an issue beyond hard problems, they did not necessarily put it in [into Bugzilla]. There are many kinds of problems [. . .] like nice-to-haves, problems with understanding things or other issues like that. (I-B-01)

Additionally, comprehensive automated logging of use data is conducted in the background with the goal of making issues reproducible. After commercial rollout, Bar's plans are to have the support as well as the marketing divisions report directly to the PO on user feedback.

2.5.2.3 Qux

Qux uses Jira and Confluence as basic infrastructure in order to scaffold agility in their development process. These tools are utilised for internal coordination, especially for the PM. Furthermore, Qux's intention is also to establish customer-facing transparency. Hence, customers can also issue tickets and feedback (depending on the project structure agreed upon with the customer).

As regards active user feedback and participation, Qux employs different methods. In some cases, customer organisations carry out their own beta testing and feedback acquisition, select and aggregate it and send it to Qux. In other cases, all data from such tests is handed over to Qux without aggregation. Another option is to rely on direct user feedback via email generated from feedback-buttons and similar options integrated into applications, without the involvement of customer organisations. Qux's employees are aware of a wide variety of tools and systems to facilitate user feedback such as TestFlight or crowdsourcing systems, but on various occasions throughout the interviews, it becomes clear that they are still searching for an optimal system, especially one that meshes with agile development and more easily supports the handling of user feedback:

> [. . .] In each release in Scrum, there is one functional area, which gets completed and released, so to speak [. . .] there's always this wish, we are looking for a suitable platform [. . .] so we can say: 'you don't have to send me an email, you don't have to write down anything, you don't have to call me [. . .] then they could just hit a button, rate it [the specific result of a sprint/release], write a short text, Twitter-style at most [. . .] which would then just be sent to us so we could look at it. (I-B-01)

Apart from users, the customer organisations themselves are also actively queried as sources for feedback. Qux's PM puts it like this:

> [. . .] obviously, we also collect feedback from our customers. When we present something [. . .], we ask them quite in a quite focused way: '[. . .] please look at this'. We have them take responsibility, which is a good thing, since basically it is their project. . . Which is why I expect them to care and not just complain in the end, after a release [. . .] They have to give feedback frequently. (I-Q-01)

Furthermore, Qux frequently employs app store reviews and ratings as feedback channels, similarly to Foo. For more qualitative evaluations regarding UUX and UI, Qux has no formal tools or channels in place. Here, they rely on a user-focused and agile company culture as described above as well as ad-hoc feedback in meetings with customers and beta testers. Internally, user/customer and/or peer feedback is not only shared via Jira but also via daily stand-ups which are emphasised as an important tool:

> [. . .] we meet at 9:00 and everybody explains what he did the day before and what he plans to do today [. . .] you don't put things off[. . .] (I-Q-03)

This ritualistic form of informal exchange is supplemented by the grapevine, as it is at Foo. Like Foo, Qux then utilises User Stories and Bugs according to Scrum to codify and work with user feedback. Both artifacts are primarily input and maintained by the PM. To be clear, the PM is not the only *source* of such data, as explained above, but he *maintains* it.

2.5.3 Filtering and Interpretation

In the previous sections, we reported how and with the participation of which roles customer feedback is gathered and passed along through Foo, Bar and Qux. There is, however, one step missing – what emerged as *filtering and interpretation.*[5] This is the process of analysing and assessing user feedback as well as matching it with other feedback and/or internal goals. It also encompasses the challenge of identifying what the user *really* means or needs.

2.5.3.1 Foo

Foo has a long history of experimenting with the incorporation of user feedback in their development cycles and within this history, there have been failures, too. One example from the interviews (I-F-01, I-F-03, I-F-05) is a former experimental project grounded in Participatory Design in which the development of a software product relied heavily on a selected group's input and co-design. The members of this group were considered lead users in their domain. However, as it transpired, the product became much too specialised and thus did not appeal to many potential customers. Experiences like this reinforced Foo's focus on filtering as well as diversifying user feedback structures – as outlined in the previous sections, customer feedback is sampled through a wide range of channels, representing an attempt to level the playing field and keep specialisation appropriate to the product. What this means is that in the example of iFin, which has a very widespread and heterogeneous

[5]This is actually an in-vivo code. A PO at Foo used those exact words.

user base, specialisation has to be kept at a much broader and shallower level than for some of Foo’s other products, e.g. those targeted at landlords and this much smaller and more focused group’s specific needs.

The most important decision makers in relation to the filtering process are the POs. They consciously try to match their vision of the product with the customer input, adapt, prioritise and, if deemed necessary, modify or reject specific feedback. These decisions are grounded in the work performed by the roles described above. The customer lab does not engage in filtering per se but rather reports comprehensively, based on proven methods. The Social Media manager engages in partial filtering – she tries to match every piece of incoming input with previous decisions made by the POs. If the input is identical or very similar, she *“informs the customer accordingly”* (I-F-02), meaning notifications such as “request denied”, “request in development” and so on. In such cases, she does not alert the POs. Should she hand a specific piece of new user feedback to the POs, she usually annotates it and states her opinion about it, i.e. actively enriches the user feedback based on her long-term experience with iFin. Notably, the support team filters actively. Customer feedback is gathered and discussed by the leader of the support team for the respective product and the chief of support, and filtered. Thus some feedback may never even reach the POs:

> If a person wants a new feature, the support employee checks the database weather the feature has already been requested by someone else. If so, the customer’s ID is added to the incident. If not, the request is inserted in the database, which triggers a message to the team leader who assesses it. If he decides that the request is useful, the entry is set to ‘visible’ for the PO and the development team. (I-F-03)

Throughout all interviews with Foo, the exact operationalisation of the filtering processes remains somewhat vague but can be categorised broadly into two classes: *Qualitative* and *Quantitative* filtering. Quantitative filtering concerns the frequency and intensity of a specific type of feedback. The support team seems to utilise quantitative filtering which is also easy to do for them since the customer feedback from each ‘call’[6] is recorded in their database. Quantitative aspects are, however, no guarantee for the feedback to get implemented – an often cited example (I01, I02, I03, I04, I05) from our interviews is that of a feature requested by a significant number of customers. However, this feature would make other, quite specific, long-term plans for the software impossible on a technical level. Hence, it is not implemented. The opposite constellation, i.e. individual or occasional cases of feedback seem more straightforward: (very) specific features which get requested by very few people usually get filtered out.

Qualitative filtering is a ‘softer’ aspect and seems primarily associated with experience, and a certain ‘artfulness’ rather than just hard data. It was next to impossible for all interviewees to really describe techniques and methods for qualitative filtering. Instead, in nearly every interview, it was stated explicitly or

[6]Terminology taken from the interviews – a ‘call’ should be understood as any kind of communication with users, not just telephone calls.

implicitly that a *lot* of knowledge about and a *"feeling for"* (I-F-01, similarly phrased also in I-F-03 and I-F-05) the product has to be developed over time in order to 'get it right'. A need to actually be a user of the product oneself has also been mentioned. All in all, Foo's Social Media management is the role with the deepest engagement in qualitative filtering procedures.

Qualitative and quantitative filtering mechanisms are reported to compliment each other well, e.g. within I-F-01 and I-F-04 and none of both aspects is viewed as sufficient by itself.

2.5.3.2 Bar

Like Foo, Bar's PO has a key function in the process of filtering and interpreting user feedback. He exclusively classifies and judges incoming information and notes from a range of roles through different channels. He has to decide and to match the pieces of information with the long-term goals for the product. On a quantitative, heuristic level, Bar judges feedback as relevant if it comes in 2–3 times in similar form:

> I give the thing [iHome] to 10 people and get 10 different opinions when I ask a specific question. That's rather difficult. At the moment, my strategy is that I look deeper into things after I hear issues 2–3 times. [. . .] Well, I always look at all the things [feedback], but when A says A, B says B and C says C, I stay with my opinion. (I-B-01)

While, so far this has proven effective, Bar's PO is also aware of the pitfalls of such an approach:

> But usually, you feel a bit like, well, the father of such a system. That makes each [. . .] feedback which is not exactly the same as your view of the system a critique and you have a certain defensive position. It is difficult to be neutral. (I-B-01)

Filtering and categorising feedback coming in from Bugzilla is managed by the DVT. They decide when and if something gets bumped up to the PO for decision making or directly to the development team for implementation. Our interview partners at Bar stated that informal, ad hoc (coffee corner-)talks between DVT, developers and the PO are central instruments in discussing, judging and triangulating feedback. Hence, we can also categorise filtering and interpretation mechanisms in Bar in quantitative and qualitative aspects. However, the structures are less complex and less differentiated than in Foo's case.

It is notable that Bar has yet to establish formal structures for who actually takes responsibility for the liaison and engagement with their test households, leading to difficulties in regards to filtering and interpreting feedback coming from those households:

> [. . .] there is the question if the developer should have frequent contact? I just don't know. Partially, sure, so he can hear opinions face-to-face and hear users' problems – just to understand. [. . .] [the users] all have their opinions. That has to be channeled in some way. Can you categorise such things? (I-B-02)

Lastly, Bar’s PO is unsure if the test users might not become blind to certain issues due to routine and debates weather the user sample should be regularly changed, at least partially:

> [. . .] you just breeze over certain issues [after engaging with the product becomes routine] [. . .] if the beta-tester is at the point [where an issue arises] for the second time, he just skips it [referring to ignoring issues or finding workarounds]. (I-B-01)

In this regard, the PO consideres the potential limitations of outsourcing feedback collection and management to a service provider. While generally managing feedback is a burden, having an external mediator between users and decision makers within Bar was also deemed problematic. The close contact with the user and the PO’s more or less direct and uncomplicated channels for approaching them for specification and further questions were valued very much, which led to the installation of the friendly user test set.

2.5.3.3 Qux

The situation for Qux is quite different from that of the other two SMEs with regards to filtering and interpretation of feedback. Based on their role as a service provider, they face the dichotomy of having to engage and negotiate with their customers as well as having to discuss and judge user and customer feedback internally. Qux’s open communication between the CEOs and the units is helpful in providing a lean and agile structure to quickly engage with such feedback. The strings of such decision making processes all converge on the PM but subsequently have to be debated with the customer who makes the final decisions, sometimes forcing Qux into less than optimal decisions:

> For the most part, it is not very good if the customer selects [test] users itself but it’s just the way it is; we don’t have the target audience on board. It’s a shame but it’s the way it is. (I-Q-03)

Yet, Qux’s employees voice rather unequivocal support for user-centered design, UUX and customer integration and – as mentioned before – reflect on those topics frequently. The Creative Director puts this in simple, decisive words:

> [Question about what would speak against a strong UCD-motivated process] “Just ignorance. If you do UCD, you put the user or the user group in your focus [. . .] which is logical. We don’t build things for animals or for little grey men but for people.” (I-Q-02)

Given their company structure, current practice in Qux’s development process is to utilise as many automated use tracking, bug-tracking and data gathering tools as possible since those can be integrated easily and quickly (and cheaply) into their products. Data can thus be gathered in the background without disturbing users. This quantitative feedback is then triangulated with qualitative information which mainly comes in through email, either directly from users or aggregated from customers. This practice meshes well with Qux’s agile process, in so far, as it can be considered during the task planning of further sprints. Quantitative heuristics similar to Bar’s case are employed to speed the process up but Qux's sentiment is that qualitative information gathered by direct exchange and engagement with users would be more valuable. However, limited staff resources as well as Qux's customer relationship make it hard to implement and also limit the possibility for Qux's staff to pose questions to users if they arise about qualitative data aggregated by Qux's customers. Trust in the validity of such data is critical but Qux has no first-hand way of ensuring this. In addition to that, handling qualitative user feedback is currently done based on the number of similar issues are reported. If things are mentioned more than twice, feedback is worth discussing further internally discussion as well as with customers.

2.6 Discussion

Based on the three contrasting case studies, we see a growing awareness of the need to integrate users into soft- and hardware development. Even a classically non-iteratively operating company such as Bar either plans user integration from the beginning or learns about the value of iterative development and feedback during the process of developing products which rely heavily on user interaction and thus on a positive UUX. Our studies also show that the number of users a company can engage with as well as the differentiation of channels and tools can scale alongside the company's size. This is problematic since small companies like Qux who perceive the need to engage more with users simply cannot do so adequately. It is certainly possible to utilise internal testing, using ad-hoc methods such as convincing friends to give feedback or put a mental emphasis on a user perspective. However, it seems that the more people actually try to do this, the more they realise that such methods are inadequate and have pitfalls such as too much introspection or blindness to certain aspects. Such concerns become especially obvious when a second web of entanglements, i.e. external customers, becomes part of the process. A possible solution to such problems might be working with external, specialised partners for certain aspects of user engagement, such as Crowd-Testing platforms or testing *as a service*.[7] Through economies of scale, such services can be offered more cheaply than building complex infrastructures for user engagement and feedback internally and might be an entry point for more in-depth work with users, as witnessed in Bar's case.

The differentiation of roles is a highly interesting factor. It, too, can – and maybe even has to – scale alongside the size of the company itself. With Foo, we have an exceptional example where, over the course of many years, a very intricate web of different roles has emerged. Those roles and their different perspectives compliment each other well and eliminate many of the insecurities and problems we saw in the other cases, such as Bar's issues with responsibilities for certain aspects of working with users or Qux's problems with data validity.

[7] An example might be Living Labs as a service, see e.g. [20].

However, different roles do not just magically compliment each other – they also clash and Foo shows how to facilitate this in a purposeful manner. Company culture is the keyword here. Foo's Social Media management might, for example, disagree *very* strongly with a PO's vision for something since she actively tries to take on a qualitatively grounded user perspective. Heated discussions can happen – and, according to Foo, they should. All roles need to be empowered enough not to fear personal or other negative consequences yet still be able to get behind the overall product vision. Agile development can help here because it facilitates constant, quick engagement within teams and provides structures in which things can be explored and tested without great risk or cost (and hence, usually personal consequences). UCD can help as well because a common denominator in developing products *for users*, not 'for oneself' can put things into perspective, especially in regards to company culture. In Qux's case, we see that even a small company can form a very strong user focus in its culture. We also see that this has significant impact on the products (if not as much as a combination with a strong base in resources and differentiated roles, tools and channels).

With regards to channels and media, careful diversification also seems advisable. Again, Foo serves as the example of a large SME but it is much less the diversity of their tools and channels but rather their project-specific usage and focus which make them successful. For example, Foo's internet forums do not require much maintenance because they are framed as 'users help users'. Foo knows that their chat option enabling users to talk to the support team is not used all that much and could be cut without too much damage should the need ever arise; but they also know that they should probably not change PO's phone numbers lest they lose the engagement with their long-term expert users who freely offer them very valuable feedback. Certainly, smaller SMEs might not reach Foo's level of diversification but they *can* and should put careful, product- and user-specific thinking into which tools and channels they actually use, how they frame them and what they do with them.

Looking at more Scrum-related channels and media, it should be noted that diversification also seems relevant here: For example at Foo, roles such as Social Media manager or the support team do not participate by default in Scrum activities such as sprint planning meetings or daily stand-ups. At Bar, we found that only software development organises itself in a Scrum fashion. On the one hand, this leaves some room for different paces (customer support, for example, simply cannot happen in sprint cycles due to its mostly reactive nature) and 'outside' perspectives.[8] On the other hand, such diversification also induces friction and can make it hard for teams or even individuals to relate to each others' work practices such as pace, focus, long-term view, technical vs. social aspects and so on. This can be moderated by establishing and fostering informal exchanges and including non-members of the Scrum team into Scrum events, tools and channels on a case-by-case basis. However, we have our doubts that there can be a one-size-fits-all template on how to do this. It is an individual, highly context-specific process that needs time, intelligent *and*

[8]As in outside of sprints and their – by definition – extreme focus.

emphatic people as well as studying examples and constantly evaluating your own approaches as well as being willing to change them. In our opinion, this is actually one of the central advantages of agile processes with regards to UCD and UUX: they cultivate a culture steeped in constant self-evaluation, iterative changes and – crucially – the fact that rollbacks can be necessary and completely acceptable.

There seems to be one central point which can make or break user-centered agile software development in a Scrum project and that is the PO. There is a very great deal of power about the agile process itself as well as about UCD/UUX aspects centralised in one single person. The demands on such a person are very high and, depending on the project, may even be too high. Hedging one's bets in the sense of utilising more than one PO might, consequently, make sense as has already been indicated in literature [27]. We see this in all our cases. It is most visible in Foo's case with three POs, but Bar and Qux also distribute some aspects of what might be construed in strict Scrum doctrine (if there is such a thing) as the PO's responsibilities. Hence, codifying strict roles might not always be advisable and a certain leeway might make sense. For example, appointing a senior member of a UX-design team to a part-time PO assisting a continuous PO might be worth considering if actually employing two full-time POs with complimenting skill-sets is not viable due to project or financial constraints. However, not all agile processes are Scrum and most development processes – in practice – are not textbook Scrum or any other process model but are rather oriented on guidelines and otherwise adapted. Yet, based on our results and our experiences as well as other published research such as [27], we would think it likely that the importance of the PO is generalisable to an extent for processes involving a role similar to a PO.

Concerning the crucial aspect of filtering and interpreting user feedback, we also would like to point to the differentiation of roles, tools and channels as well as to a solid company culture as the main factors for success. Furthermore, making conscious decisions about including qualitative as well as quantitative types of data in the development process seems highly advisable as well as economic. Regarding the operationalisation of filtering and interpretative techniques; quantitative filtering seems to be the more straightforward one: given thorough documentation in database form, user feedback can be quantified and analysed rather easily. This data can supply very valuable intelligence into trends. However, it seems extremely important to supplement the quantitative view qualitatively: a *good* idea is not necessarily the same as an *often requested* one. This makes qualitative filtering a necessity. To use an analogy, in our interviews, we found certain similarities between this kind of filtering and qualitative sciences like ethnography: deep immersion into a product's user base and using the product oneself – getting a *feel* for it and forming *experience* – has been stated as very important and again, different perspectives and their intersections are considered valuable (one could compare this to the concept of inter-coder reliability in qualitative data analysis). Furthermore, a certain distance from possible moderating factors (like budget aspects or other business influences) seems associated with successful qualitative filtering in a manner not unlike the (artificial) naive approach utilised by ethnomethodologists. All in all, both views compliment each other and if possible, neither should be viewed on its own when engaging in filtering and interpretative action.

Thinking into the future, HCI and CSCW might provide help on the intersection of agile development and UCD in certain areas: As indicated by Qux's wishes for lean, almost Twitter-style feedback tools, Bar's utilisation of a partly externalised user testing infrastructure or Foo's quick and easy in-house tool to work with app store reviews, properly (co-)designed tools to support agile and user-centered processes are lacking. There are concepts from HCI and CSCW, such as comprehensive situated user feedback and engagement mechanisms right inside of products, see e.g. [34] or leveraging modern mobile devices to facilitate relatively lean, event-contingent qualitative and quantitative data collection [7]. However, even if we as researchers might not necessarily like it, those concepts can sometimes be unwieldy and are not necessarily suitable for market-driven environments, necessitating collaboration between researchers and professionals.[9]

2.7 Conclusion

We believe that there is no one-size-fits-all template for UCD/UUX and agile software development. The integration of user centered and agile principles is an artful business which necessitates many case-by-case decisions. However, we also believe that case studies such as the ones described in this contribution can help navigate at least parts of this difficulty – which, incidentally, is also why we decided to keep our discussion on a relatively high level. Furthermore, we think that *some* principles might be abstracted and generalised. We would like to close this contribution with a presentation of those principles by way of a concise section on suggestions.[10]

2.7.1 Suggestions for Integrating UCD and Agile Software Development

UCD is important for good UUX (and market success): This is the most obvious point and well-established in the scientific community but given the fact that multiple SMEs do not yet focus on UCD, it needs to be re-iterated.

[9]An attempt at an explicitly simple and lean user feedback system similar to what Qux wished for is currently being developed open source led by our research group. It is called 'Shake' and interested parties are welcome to try it out and/or contribute on http://github.com/UniSiegenCSCW/Shake.

[10]However, please keep in mind that those suggestions are grounded in literature and three essentially qualitative case studies. They can make no claim to completeness or applicability in *all* but we believe they are helpful in *many*.

Agile culture: Agile principles such as quick and iterative work (as well as scrapping things if need be) are well suited to be interwoven with UCD methods and user integration. However, there are still open questions (see below) and each case is *individual*.

Company culture: Multiple people with multiple perspectives need to have voices in the process and should to be able to challenge decision-making processes without retribution. Space and opportunity for informal exchanges and the grapevine are vital.

Differentiate roles, channels and tools: A differentiated, yet holistically considered organisational structure is a necessary base for UCD and agile development and should be constantly iterated upon. User integration at (too) isolated points might even be counter-productive. Triangulation is necessary.

Filtering and interpretation are necessary: Not everything that a customer wants can be done or is actually a good idea and vice versa. Good ideas can be hard to come by. Qualitative and quantitative filtering mechanisms should be employed.

Filtering is not trivial: Staff need to be educated, to actually use the product and to develop an appropriate frame of mind. Supportive ICT systems can be useful but are not necessarily available.

The PO is the critical point: A PO needs to make a significant amount of highly relevant decisions, which is why the person filling such a role needs a grounded (multi-stage) base for those decisions and a quite comprehensive skill-set.

Consider more than one PO: It may be sensible to employ more than one PO or at least to treat the role more fluidly. If this is done, it is vital to establish and communicate the different responsibilities of the POs so not to impact the agile process negatively.

Acknowledgements We thank all employees of three SMEs who participated in this study. This contribution has been funded by the German BMWi (Federal Ministry for Economic Affairs and Energy) though the projects CUBES (FKZ: 01MU14001A) and SmartLive (FKZ: 01MU12026A).

References

1. Anderson DJ, Reinertsen DG (2010) Kanban: successful evolutionary change for your technology business. Blue Hole Press, Sequim
2. Beck K, Beedle M, Van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin RC, Mellor S, Schwaber K, Sutherland J, Thomas D (2001) Agile manifesto. http://agilemanifesto.org/
3. Beyer H (2010) User-centered agile methods. In: Carrol JM (ed) Synthesis lectures on human-centered informatics, vol 3. Morgan & Claypool Publishers, pp 1–71
4. Bratteteig T, Bjerknes G (1995) User participation and democracy: a discussion of Scandinavian research on system development. Scand J Inf Syst 7(1):73–98
5. Braun V, Clarke V (2006) Using thematic analysis in psychology. Qual Res Psychol 3:77–101
6. Chamberlain S, Sharp H, Maiden N (2006) Towards a framework for integrating agile development and user-centred design. In: Extreme programming and agile processes in software engineering, Oulu, vol 4044, pp 143–153

7. Dax J, Ludwig T, Meurer J, Pipek V, Stein M, Stevens G (2015) FRAMES – a framework for adaptable mobile event-contingent self-report studies. In: Diaz P, Pipek V, Ardito C, Jensen C, Aedo I, Boden A (eds) End-user development. Lecture notes in computer science, vol 9083. Springer, Cham, pp 141–155
8. Draxler S, Stickel O, Winter D, Stevens G (2014) Nutzerintegration in softwareprojekte durch multi-channel feedback. In: Butz A, Koch M, Schlichter J (eds) Mensch & computer 2014 – Tagungsband. De Gruyter Oldenbourg, Berlin, pp 175–184
9. Ehn P, Kyng M (1987) The collective resource approach to system design. In: Kyng M, Bjerknes G, Ehn P (eds) Computers and democracy: a Scandinavian challenge. Avebury, Brookfield, pp 17–57
10. Ferreira J, Noble J, Biddle R (2007) Agile development iterations and UI design. In: Proceedings of the AGILE 2007, AGILE '07. IEEE Computer Society, Washington, DC, pp 50–58
11. Floyd C, Mehl WM, Reisin FM, Schmidt G, Wolf G (1989) Out of Scandinavia: alternative approaches to software design and system development. Hum-Comput Interact 4(4):253–350
12. Floyd C, Reisin FM, Schmidt G (1989) STEPS to software development with users. In: ESEC '89: proceedings of the 2nd European software engineering conference. Springer, London, pp 48–64
13. Hansson C, Dittrich Y, Randall D (2006) How to include users in the development of off-the-shelf software: a case for complementing participatory design with agile development. In: Proceedings of the 39th annual Hawaii international conference on system sciences, HICSS '06, Kauai, vol 8, pp 175c–175c
14. Hering D, Kraft X, Schwartz T, Wulf V (2013) Usability-Hindernisse bei Software entwickelnden KMU. In: Boll S, MaaßS, Malaka R (eds) Mensch & computer 2013 – Workshopband, pp 9–18. Oldenbourg Verlag, München
15. Holtzblatt K, Beyer H (1993) Making customer-centered design work for teams. Commun ACM 36(10):92–103
16. Isomursu M, Sirotkin A, Voltti P, Halonen M (2012) User experience design goes agile in lean transformation – a case study. In: 2012 agile conference, Dallas, pp 1–10
17. Keiningham TL, Cooil B, Andreassen TW, Aksoy L (2007) A longitudinal examination of net promoter and firm revenue growth. J Market 71(3):39–51
18. Larusdottir M, Cajander A, Gulliksen J, Cockton G, Gregory P, Salah D (2014) On the integration of user centred design in agile development. In: Proceedings of the 8th Nordic conference on human-computer interaction, NordiCHI '14. ACM, New York, pp 817–820
19. Lee JC (2006) Embracing agile development of usable software systems. In: CHI '06 extended abstracts on human factors in computing systems, CHI EA '06. ACM, New York, pp 1767–1770
20. Ley B, Ogonowski C, Mu M, Hess J, Race N, Randall D, Rouncefield M, Wulf V (2014) At home with users: a comparative view of living labs. Interact Comput 27:21–35
21. Lievesley MA, Yee JSR (2006) The role of the interaction designer in an agile software development process. In: CHI '06 extended abstracts on human factors in computing systems, CHI EA '06. ACM, New York, pp 1025–1030
22. Muller M, Haslwanter J, Dayton T (1997) Participatory practices in the software lifecycle. In: Helander M, Landauer T, Prabhu P (eds) Handbook of human-computer interaction. Elsevier, Amsterdam/New York, pp 256–297
23. Pipek V, Wulf V (2009) Infrastructuring: towards an integrated perspetive on the design and use of information technology. J Assoc Inf Syst 10(5):447–473
24. Reichheld FF (2003) The one number you need to grow. Harv Bus Rev 81(12):46–54
25. Schwaber K (1995) SCRUM development process. In: Proceedings of the 10th annual ACM conference on object oriented programming systems, languages, and applications (OOPSLA), Austin, pp 117–134
26. Silva T, Silveira MS, Maurer F, Hellmann T (2012) Paulo: user experience design and agile development: from theory to practice. J Softw Eng Appl 5:743–751

27. Singh M (2008) U-SCRUM: an agile methodology for promoting usability. In: Proceedings of the agile 2008, AGILE ’08. IEEE Computer Society, Washington, DC, pp 555–560
28. Strauss A, Corbin J (2008) Basics of qualitative research grounded theory procedures and techniques. SAGE Publications, Los Angeles
29. Sy D (2007) Adapting usability investigations for agile user-centered design. J Usability Stud 2:112–132
30. von Hippel E (2005) Democratizing innovation. MIT, Cambridge
31. Williams L, Cockburn A (2003) Agile software development: it’s about feedback and change. Computer 36(6):39–43
32. Wulf V, Rohde M (1995) Towards an integrated organization and technology development. In: Symposium on designing interactive systems (DIS’95). ACM, Ann Arbor, pp 55–64
33. Wulf V, Rohde M, Pipek V, Stevens G (2011) Engaging with practices: design case studies as a research framework in CSCW. In: Proceedings of the ACM conference on Computer supported cooperative work. ACM, New York/Hangzhou, pp 505–512
34. Yetim F, Draxler S, Stevens G, Wulf V (2012) Fostering continuous user participation by embedding a communication support tool in user interfaces. AIS Trans Hum-Comput Interact 4(2):153–168