Chapter 6
Communication Breakdowns in the Integration of User-Centred Design and Agile Development

Silvia Bordin and Antonella De Angeli

Abstract Despite several calls for a more systematic integration of User-Centred Design and Agile development methodologies, no satisfactory agreement has been found yet. We articulate three breakdowns that may occur when integrating these two software engineering approaches, namely a variable interpretation of user involvement, a mismatch in the value of documentation, and a misalignment in iterations. These themes emerged from theoretical grounding as part of action research in a case study where UCD and Agile were integrated to develop mobile applications for a user community. We discuss attempted strategies for improving community involvement alongside the evolution of the project team, composed of developers, designers, users, and customers. We finally suggest ways to promote a receptive organisational culture for the integration of UCD and Agile, drawing inspiration from participatory design and design thinking, retaining the richness of community voice, and effectively timing the combination of the two methodologies.

Keywords Working practices • Organisational culture • Research into design

6.1 Introduction

In recent years there has been a growing interest in understanding the combination of the user-centred design (UCD) and Agile development approaches, both in the human-computer interaction community and in the software engineering one. This convergence would lead to a more holistic software engineering approach relative to the application of one of the individual methodologies alone [1]. The advantage is twofold: on the one hand, Agile methodologies do not explicitly address usability or user experience (UX) aspects in their understanding of the development process, although valuing customer satisfaction [2, 3]. Yet, these aspects cannot be overlooked anymore, since a carefully designed UX can provide an advantage over competing products [4]. In fact, despite its problematic nature

S. Bordin (✉) • A. De Angeli
Department of Information Engineering and Computer Science, University of Trento, Trento, Italy
e-mail: bordin@disi.unitn.it; antonella.deangeli@disi.unitn.it

and high costs [5], effective user involvement results in high reward and several expected benefits including improved quality, understanding, and acceptance of the product [5], and generally overall "positive effects on both system success and user satisfaction" [6].

On the other hand, UCD does not explicitly address how the implementation of the design should be performed, despite needing to ensure that no "design drift" [1] occurs by maintaining a tight connection with the development. Agile methodologies appear to fill this gap because of their capability to flexibly respond to change in requirements, priorities and context, benefitting from a constant involvement of the customer in the process. These features have facilitated their widespread adoption in companies [7]. Furthermore, the intrinsically iterative nature of Agile implies continuous testing and incremental improvement of stable versions of a software product, fostering an overall higher quality of the final outcome, and aligning in principle with the iterative nature of UCD.

Despite these benefits, no satisfactory way to integrate UCD and Agile has become established yet, if one exists. We contribute to filling this gap by reflecting on our own experience as researchers and designers in a project where the integration of UCD and Agile was used in mobile application development, not for a single business customer, but for a whole community of users. By intertwining research into design [8] and a literature review, we identified three communication breakdowns that are likely to occur because of a mismatch in the formalisation of key themes in the two approaches, namely the interpretation of user involvement, the value of documentation, and the synchronisation of iterations. We believe that the analysis of such communication breakdowns can provide a distinctive analytical tool to facilitate a shared understanding of the respective assumptions of UCD and Agile and of the potential conflicts between them, thus supporting new kinds of collaborative work arrangements.

Consistently with [9], we observed that all breakdowns are manifested at the work process level, but that their solution requires changes in the organisational structure. As a result, we propose some reflections on the need to foster the establishment of a suitable and receptive organisational culture in order for an effective integration of UCD and Agile to occur; these considerations are inspired by the participatory design and design thinking approaches. We also reflect on how to retain the articulation of the needs and positions expressed by a whole user community and, in a more process-oriented perspective, on which is the most suitable timing to integrate UCD and Agile, taking into account the peculiarities of the two methodologies.

6.2 State of the Art

User-centred design is an umbrella term used to denote a set of techniques, methods, procedures and processes that places the user at the centre of an iterative design process [10]. As stated by Norman already 30 years ago, when UCD was struggling to establish itself in a predominantly technology-pushed environment, "the purpose of the system is to serve the user, not to use a specific technology, not to be an elegant piece of programming" [11]. Agile also presents a variety of methods advocating a lightweight approach to software development where rapid and flexible adaptation to change is the key to maximising customer satisfaction. Such adaptation is achieved through a process of continuously improving, evolutionary development carried out by a self-organising, cross-functional team that communicates through face-to-face and often informal meetings rather than through formal documentation. Agile methodologies are grounded on the principles and values listed in the Agile Manifesto [12], which, among other things, highlights the concept of customer collaboration: the customer is in fact expected to be actively involved in the development process, although to a variable extent depending on the specific form of the methodology in practice, and to have the power to steer the direction of the project by intervening on the requirements according to his/her possibly changing needs [13].

We highlight that both UCD and Agile can be instantiated in a large variety of practices: this makes it difficult to present a definitive position on the integration of the two approaches. However, in the next sections we will discuss three themes in particular that may have diverging interpretations in the two methodologies, namely user involvement, documentation, and synchronisation of iterations. These themes, which emerged from a case study and were then corroborated through a literature review, are most likely to cause communication breakdowns, i.e. examples of "disruption that occurs when previously successful work practices fail, or changes in the work situation (new work-group, new technology, policy, etc.) nullify specific work practices or routines of the organizational actors and there are no ready-at-hand recovery strategies" [9]. Communication breakdowns occur in the absence of a shared meaning among team members and can be counteracted by designing for translucence (i.e. the combination of visibility, awareness and accountability); shared meaning is nurtured by grounding (the communication process aimed at the creation of mutual knowledge, assumptions, and beliefs [14]) and consolidated by knowledge sharing.

6.2.1 Integration

A growing interest in understanding the combination of UCD and Agile is emerging, as witnessed for instance in the literature reviews performed by Jurca [4] and Salah [1], which consider 76 and 71 papers respectively. The papers were selected based on their focus on methodologies for integrating UCD and usability engineering with Agile approaches, with an accent on the software engineering community, to which the authors belong. Results suggest that much literature on the topic consists of "experience reports", with "few rigorously conducted studies in Agile-UX" [4] or systematic guidelines. Moreover, that research is dispersed over a large number of venues and is likely to not fully reach its intended target community. Both studies conclude that there is a need for design and development methodologies that can "draw on the best practices and tools of the two disciplines" [3].

Attempts at integrating the practices of UCD and Agile development have been made over time [e.g. 15], leveraging the similarities of the two methods while mitigating their differences. An approach called *Agile usability* is proposed in [16], enriching the Extreme Programming methodology with several artefacts drawn from HCI each used in different moments. Similarly, some experts suggest applying discount usability methods by involving a very small number of users (one to three), with frequent testing and with constant updates to the team and the clients [17]. Others propose to lighten UCD approaches in order to keep the pace of Agile iterations [18]: for example, the presence of an onsite customer is reported as a common practice in Agile projects to facilitate the communication of requirements to developers [3]. Ungar and White [19] describe a case study in which the application of the design studio approach might effectively bridge UCD methods and Agile ones, such as Scrum: this approach envisages a rapid, iterative process of ideas generation, discussion, and reconciliation into a unique design concept to be implemented. Lean UX [20] is a practitioners' proposal for the integration of UCD practices into Agile; yet, as argued in [21], the organisation applying this methodology has to be ready for it, developing an appropriate internal culture, which is often a non-trivial condition to achieve.

While acknowledging the large and robust common ground [14] that the two approaches anyway share, in the next paragraphs we will introduce three communication breakdowns [9] that are likely to affect their integration.

6.2.2 User Involvement

The concept of user-centredness is complex and covers a broad range of perspectives, which according to [22] are articulated around four dimensions: user focus, work-centredness, involvement or participation, and system personalisation. Different and at time conflicting interpretations of processes, practices, and goals exist within these macro-areas. The dimension of involvement is particularly interesting for our work as it describes different approaches related to the user role in design, ranging from an *informative* role (users act as providers of information and as objects of observation) to a *consultative* role (users comment on predefined design solutions) to a *participative* role (users actively take part in the design process and have decision-making power regarding solutions) [5]. User involvement in the informative/consultative role stresses a functional empowerment of users in its focus on designing usable and satisfying systems; this differs from the participatory design perspective, which aims at a democratic empowerment [23] by allowing people to shape the tools which will affect their work or personal life.

In UCD, user-centredness is typically described in this continuum between involvement and participation [22]: user needs and activities are thoroughly researched and understood by the design team upfront or in direct collaboration with a small sample of selected users. Agile techniques also encourage the customer's collaboration throughout the development process; however, in the Agile terminology, the notion of customer is often blended with that of a user [24], with contrasting interpretations of the distinction between these two concepts and of whom is supposed to take this role. Most Agile methods in fact define the customer as a representative of the end users who has direct and regular contact with them [e.g., 2, 25, 26]. However, some authors report that it is infrequent for a real end-user to act as a customer [15, 27], subsequently questioning the extent to which the customer can actually represent the real user and his/her needs [3, 28]. Others [15] recommend that the customer's engagement should also be supported by a number of other roles within the team, such as a proxy user, or that the customer role should be filled by one or more members of the product team [29], since customers are part of the release planning and iterative development process [28]. The duties of the customer, in their fuzzy definitions, include acting as the voice of the end user, evaluating performance and helping to prioritise and plan cycles and releases [30]. This responsibility can turn out to be overwhelming [25] and some authors argue that there is no guidance on how the customer should be able to articulate his/her needs in order to communicate the requirements to developers [28, 31].

Indeed, the very same capability of users to articulate their own working practices or to design a system can be questioned [28]; furthermore, because of a mutual learning effect, some authors claim that the more the "representative" customer becomes part of the development team, the less useful he/she is as a user surrogate [28, 32]. The same authors also propose a distinction between the user and the customer: the user interacts with the system being designed directly and uses it to accomplish his/her job, while the notion of customer is broader: understanding the users is needed to achieve a good design, understanding the customer is needed for its acceptance.

In general, given all these considerations, it seems that the integration of user needs within the feature-oriented Agile development process has not been fully achieved yet; as concluded in [3], one of the reasons is "a lack of tactics and practices" within organisations.

6.2.3 Documentation

Independently of the key issue about who is expected to be a team member (developers, designers, usability experts, users and/or customers) in an integrated UCD/Agile project, both methodologies place an emphasis on frequent communication among team members to support project awareness. However, while UCD has produced a number of design tools to support communication with developers, such as scenarios or personas, Agile tends to emphasise face-to-face informal communication. In a UCD process, formal documentation may record design rationales, list user and interface requirements, and provide the ground truth about the overall design vision, becoming "crucial for estimation and implementation efforts" [10]. Therefore, the experts will devote an important amount of time to analysing users and their tasks and then iteratively collecting feedback; these activities need to be performed and documented before the implementation phase begins.

Conversely, in Agile development the use of documentation is diminished [33], to the point that one of the principles of the Agile manifesto [12] states that "the most efficient and effective method of conveying information to and within a development team is face-to-face conversation". In [34], three types of collaborative work to realign designers and developers are identified: all of them are oral and the use of documentation is not even mentioned. Because of this, rather than having requirement documents, the Agile approach incorporates the user (or his/her representative) directly in the development team. There is anyway an on-going discussion within the Agile community concerning the fact that documentation should not be discarded altogether, especially given the complexity of modern systems, and that it just needs to be adapted to more dynamic processes [35]: the argument is hence about what is to be documented [35], how, and for what purposes (e.g. supporting organisational memory and communicating with stakeholders [36]). Nonetheless, often usability goals are documented in a very general way, relying on an oral common understanding instead [7]: this may however make a quantitative evaluation of such goals problematic and make the fulfilment of the "big picture of UX" more difficult [37].

6.2.4 Synchronisation of Iterations

One of the generally accepted principles of UCD defines it as iterative [38]; Agile development is instead intended as being not only iterative, but also incremental [30]. A further challenge is about how to synchronise the periods of UCD and Agile [1, 4], and in particular whether the two methodologies should proceed in parallel or not. Several proposals envision designers and developers working closely together in a synchronised manner. For example, in [39] a daily interaction between them is defined as "essential" for a successful outcome of the project; in [34], their collaboration is defined as informal, oral and ad-hoc. Schwartz [31] found that the development pace was better maintained in a project with a usability expert than without one: in addition, the former situation gave rise to *pair designing*, in which the developer and the usability expert worked together and learned from each other, thus improving HCI practices and knowledge in the whole team and in general resulting in a better project dynamic.

Other researchers propose to keep UCD and Agile separate instead, while just synchronising their periods of iteration [15] so that design stays ahead of development. This is the interpretation of Agile usability given by Nodder and Nielsen [40], who describe a process where design and development belong to two parallel tracks and the former feeds the latter with progressively refined user requirements, prototypes and tests. Similarly, in [29] the author recommends that the lengths of the iteration of the design and development tracks have to be carefully balanced so as to allow some advance for the design activities. Another successful example of dual-track approach is described in [39]: the author reports that, in order to accommodate the different paces of UCD and Agile, the timing and frequency of data collection, rather than the methods, changed considerably.

A distinct issue concerns the amount of work, and specifically design, to be performed before the implementation phase begins [41]. UCD encourages the team to understand their users as much as possible before writing code [15]: experts will devote an important amount of time to analyse users and their tasks in order to create the interface specification document and then to iteratively collect feedback from users through various usability techniques. Despite being time-consuming activities, data collection and analysis are nevertheless considered to be necessary to inform implementation. Conversely, given its feature-oriented nature and the emphasis put on early software delivery, Agile methods are largely against an up-front period of investigation at the expense of writing code [15]: they capture "user stories" [24] instead, that is high-level requirements to be addressed at the beginning of each iteration, therefore reducing upfront work to a minimum. Still, several authors advocate a solid understanding of the user [28] also in terms of time and effectiveness and suggest the relevance of an "Iteration 0" (e.g. [1, 29]) in which upfront design and requirement elicitation are carried out, with some degree of compromise about the duration of such activities, in order to ensure the establishment of a holistic design vision that can be shared within the team [37].

Related to this, we underline that Agile approaches are prone to focusing on the details at the expenses of the overall project vision [1]. In order to mitigate this, the responsibility for carrying the UX vision forward should be explicitly shouldered by the management and the organisational context [7, 37].

6.3 The Smart Campus Project

A large set of observations on the potential and challenges of the integration of UCD and Agile development derives from our experience in the Smart Campus project, where the two methodologies were applied to mobile application development for a community of users. Smart Campus started in 2012 in the context of establishing a living lab in the Province of Trento and lasted 3 years. The University campus was selected as the playground to experiment with a vision emphasising the role of the community as builder of services. The project aimed at creating an ecosystem that could foster students' active participation in the design and development of services for their own campus [42]. A service infrastructure was the main technological outcome of the project [42–44]; on top of that, a set of eight mobile applications (Fig. 6.1) was developed to help students with a variety of professional (tracking university achievements; managing email), social (creating and managing groups; getting information about events in the city), and private tasks (travelling through the city; keeping a multimedia diary; receiving information about the university cafeterias) [45].

UCD and Agile were chosen as useful methodologies for a project that needed a fast delivery of the product while ensuring a focus on user needs [46]: the former was applied to interface design, the latter was used to build the service architecture. The project team consisted of approximately 25 members, including interaction designers and software engineers. Furthermore, several groups of students were

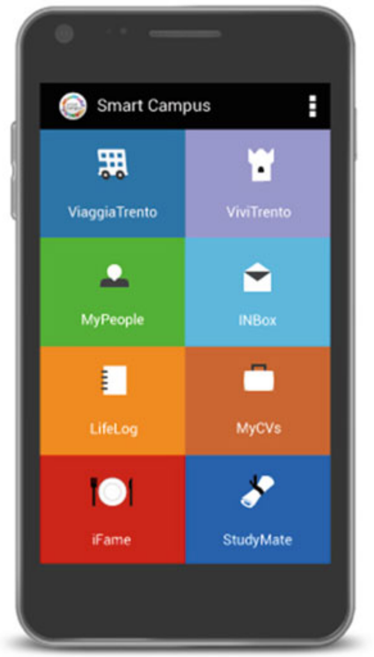

Fig. 6.1 The Smart Campus mobile applications set

involved, reaching over 500 people in total; they played the role of users or customers at different points of the project. Approaches for community engagement changed as the socio-technical infrastructure was evolving, as discussed in [42]. Some of these students were directly included in the living lab as interns, in a participatory development effort, while others played a consultative role commenting on the applications as they were developed. The dialogue between developers, designers and users was mediated by a set of communication channels including a forum (based on *phpbb*, a simple open source bulletin board system, a beta testing community in Google Play, and social networks (i.e. Twitter, Google+, Facebook, LinkedIn). In an effort to promote the sustainability of the project [47], the code was released as open source on GitHub, a platform for collaborative development.

6.3.1 Project Methodology

During the first year of the project, design and development proceeded on two parallel tracks: design focused on conceptual work and community engagement, while development focused on the service infrastructure. Several design activities such as focus groups, diaries, online ethnography and workshops were put in place, engaging 60 bachelor and master students overall; these activities aimed to investigate the life of the student population and understand the design space

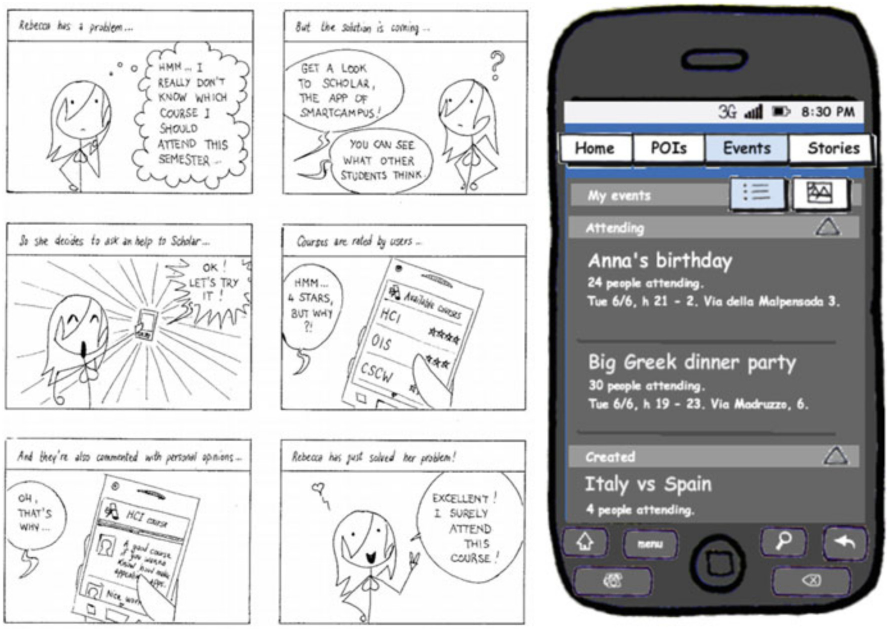
Rebecca has a problem...
HMM... I REALLY DON'T KNOW WHICH COURSE I SHOULD ATTEND THIS SEMESTER...
But the solution is coming...
GET A LOOK TO SCHOLAR, THE APP OF SMARTCAMPUS!
YOU CAN SEE WHAT OTHER STUDENTS THINK
?
So she decides to ask an help to Scholar...
OK! LET'S TRY IT!
Courses are rated by users...
HMM... 4 STARS, BUT WHY ?!
Available courses
HCI
OIS
CSCW
And they're also commented with personal opinions...
OH, THAT'S WHY...
HCI course
Rebecca has just solved her problem!
EXCELLENT! I SURELY ATTEND THIS COURSE!
3G 8:30 PM
Home
POIs
Events
Stories
My events
Attending
Anna's birthday
24 people attending.
Tue 6/6, h 21 - 2. Via della Malpensada 3.
Big Greek dinner party
30 people attending.
Tue 6/6, h 19 - 23. Via Madruzzo, 6.
Created
Italy vs Spain
4 people attending.
menu

Fig. 6.2 Different examples of low-fidelity prototypes for the Smart Campus applications

of the project. At the same time, we also performed a benchmarking of mobile apps offered by other universities and studies of online student communities. This information was used to build personas, scenarios, and later storyboards and sketches that informed the vision of Smart Campus as a toolbox containing separated but interrelated services for students' use (Fig. 6.2).

In parallel, the backend functionalities of the service platform began to be developed following a rather traditional, incremental, and non-Agile approach. The apps were then released to a growing community of campus students, starting from 90 students on the Human-Computer Interaction course for bachelor students within the local Department of Computer Science. This first group of users was also provided with a smartphone and a paid data plan in order to ease the testing, as few of them had a suitable device and we aimed to seed a user community [42].

These initial UCD and development techniques required both time and documentation. However, as the technological infrastructure became more mature and demands for more frequent app releases became more urgent, the management realised that a much faster development pace was needed: therefore, almost a year into the project, the development team quickly transitioned to an Agile development methodology, namely Scrum. Since the team had little experience with this approach, the CTO read manuals about how to implement it, involving the management in this training as well, and then briefly introduced the Agile practices to the team during a sample sprint planning meeting. Scrum was not applied by the book, but it required accommodating the peculiarities of the existing team: for instance, the CTO also took the role of the Scrum master, while the rest of the project management kept prerogatives such as maintaining relationships with the University and administrating personnel and funds; appointed “champions” of the apps, i.e. members of the development or management who were responsible for a specific app and in charge of tracking its progress, became different product owners; interns were incorporated as on-site user representatives (at least in principle, as we will see later).

The transition to Agile disrupted the alignment between the design and the development team: even though they were both following iterative approaches, it soon became clear that the iterations required by UCD were longer than those envisaged by the Scrum sprints, which were set to last 1 or 2 weeks. In fact, the design team was supposed to be ahead of the development team by at least one cycle [48], in order to be able to transfer prototypes and requirements to the development team in a timely manner. At the same time, however, designers had to face a large amount of qualitative feedback continuously coming from the user community, analyse it, and prioritise extracted requirements. A ticket was created in an internal system for most relevant user suggestions; designers would prepare design solutions and then pass them on to the developers for the implementation.

Overall, the project needed to find a suitable collaboration protocol between the design and the development team that could effectively accommodate the feedback coming from the community while not impeding the development pace. To this end, we tried several approaches to maximise team communication, which for our convenience will be classified based on their methodological structure as *formal*, *semi-formal*, and *informal* approaches.

The Formal Approach The formal approach was encouraged by the management and consisted of passing on any design issues and decisions in a written manner: this conflicted with the prescriptions of the Agile methodology. The first attempt was to use an internal wiki for collaborators, where a table of issues to be solved was maintained: this method was good for keeping track of all the problems encountered, but was not a self-explanatory procedure, as it often required additional information to be provided through different media as shared documents. The wiki was eventually abandoned in favour of maintaining a presence on GitHub, in order to promote open source contributions to the project and encourage the interventions of the community directly related to the code. This approach proved somewhat restrictive for designers as GitHub is used mostly to report bugs or issues with software behaviour, but it is not meant to support the tracking of progress about the overall UX design or the usage by people without technical expertise.

The Semi-Formal Approach In the semi-formal approach, we include different kinds of meetings. Notably, none of them directly involved any user representatives with the exception of the students involved in the Smart Campus development lab, who were actively contributing to a participatory development approach. *HCI meetings* usually gathered the whole design team and the champions of the apps. *Matchmaking meetings* were usually held once every 2 weeks and involved all of the Smart Campus staff (managers, developers, interns and designers): during these meetings, project landmarks and dates were discussed, problems brought up, and work divided. *Scrum meetings* basically replaced matchmaking ones when the team adopted an Agile approach. They were quick daily meetings, usually held early in the morning, aimed at checking the status of the project and estimating how a specific task was progressing. In turns, the members of the group involved in the Scrum reported about any problems they encountered, tasks performed and plans for the day. The Scrum Master coordinated the activity by annotating this information on a progress chart to check the status of the work and the feasibility of set goals.

The Informal Approach The informal approach consisted of face-to-face meetings between a developer and a designer, often resulting in pair designing, i.e. in the two working together in close collaboration to modify and improve the user interface during the sprints. In line with the Agile development spirit, these meetings produced no documentation. Similarly, at times developers engaged in quick chats over instant messaging systems such as Google Hangouts or Skype. In general, this approach was applied to solve specific user interface issues, not to address overarching UX themes such as for instance transitioning to the most recent look and feel suggested by the Android design principles.

The Mixed Approach Mixed approaches, between semi-formal and informal, were used in specific situations like the so-called *crazy weeks*, i.e. accelerated sprints where the whole team concentrated its efforts in order to reach a mutually agreed goal. This method was used two or three times in a year, usually to enhance aesthetics and functionalities when envisioning an immediate release of the apps. At the beginning, developers, managers and designers met to discuss on "show-stoppers", i.e. issues that would seriously compromise the usability of the app and would not allow its release; each issue could include different tasks, such as prototyping and development, and was usually assigned both to a designer and a developer, in a *pair designing* effort.

Despite this wealth of attempted collaboration strategies, the project team felt that none of them was actually fully satisfactory in integrating design and development activities. As problematic situations emerged in the project, we referred to literature in order to understand whether the issues were specific to the project or whether they had also been encountered elsewhere. We therefore combined two complementary approaches to research into design [8]: a bottom-up one, aimed at extracting themes from the analysis of project data, and a top-down one, aimed at consolidating such themes through a substantial literature review.

6.3.2 Data Analysis

The data reported in this work were collected through a number of different sources, including formal interview studies involving team members and the student community, and personal observations of the authors, who participated in the project respectively as an interaction designer and the principal investigator. In claiming the value of the findings derived from personal observations, we refer to the concept of “autobiographical design” [49]: while its authors define it as “design research drawing on extensive, genuine usage by those creating or building the system”, we intend autobiographical data as drawing on the same kind of usage, but this time by those creating or building *practices* rather than a system.

A first interview study was performed in summer 2013 after an intense period of Agile development to investigate the perception of user involvement in the project, also focusing on issues of team coordination and awareness. It engaged 20 people: 7 staff members (developers and HCI researchers) and 13 students with different levels of involvement with the project. A second study was carried out in spring 2014, focusing on the project documentation and the practices to create and use it. Among the 12 interviewees, 7 were developers in the Smart Campus lab, while 5 were students involved in the participatory development activities.

To decrease possible social bias, all interviews were conducted by a researcher external to the project. Audiotapes were transcribed and iteratively coded by the authors through thematic analysis [50]; double coding was performed on 25 % of the transcripts yielding an inter-rater reliability of 93 %. Interview analysis was conducted after the end of the project, in parallel to the literature review previously presented. Citations in the next paragraphs will be attributed to interviewees as follows: *Dev* for developers; *Des* for designers; *Int* for interns; *Stud* for students in the community.

6.4 Results

This section is composed of three parts focusing on the communication breakdowns previously introduced. For each of them we articulate the strategies employed in the project and discuss their perception according to the team.

6.4.1 User Involvement

The user and the customer were clearly differentiated in Smart Campus; we fully embrace the vision outlined for instance in [15, 27], according to which these figures denote different targets of design interventions. The customer of the project was the local University: in line with the funding scheme of the project, they contributed the case study and some personnel time. Most of the allocated personnel were appointed either from high-level administrative managers in charge of Education and IT, who participated in formal meetings with the project team, or from academic staff, who supported the research work leveraging on a richer variance of communication contexts by keeping in close contact with the Smart Campus team. The relationship with the customer was always complex and led to a partial dismissal of the project result after the ending of the financial support from the funding body. The fact that the University is no longer sponsoring the Smart Campus apps as we write is a clear sign that the communication with the customer failed during the project, reinforcing Bjørn and Ngwenyama's considerations [9] about the organisational structure being the main responsible for the failure or success of working practices. However, the discussion of this topic is outside the scope of this chapter.

User involvement in the Smart Campus project ranged along the continuum between involvement and participation [22] and evolved over a period of 3 years, where we attempted to transform our users (the students in the campus) into a community of service developers [42]. Students were in fact involved in the project initially with an informative role, for example through questionnaire and diary studies, then with a consultative role when they acted as beta testers of the apps developed by the lab, and finally as participants in the development of their own apps when integrated as interns in the project team. This process was successful in that it delivered a set of eight mobile apps, two of which entirely designed and developed by the student community, which were adopted in everyday life often with positive evaluations. Despite the convenience of our user target, especially as both authors fulfil educational duties in the local University campus, the relationship with the students often led to difficulties and to the need of reconciling the meaning of user involvement between the designers and the developers. The communication with the students therefore spanned through the continuum of our interventions, from a formal approach (when students were required to evaluate the apps as part of a coursework) to a very informal one, as in frequent corridor conversations or short text messages.

During the project we highlighted a variable and at times contrasting perception of user involvement between designers and developers. In particular, the typical breakdowns occurring in the integration of UCD and Agile were exacerbated by what the community role actually entailed: this became evident at a management meeting in April 2013. The discussion regarded the case of a student who was particularly active in the forum and willing to contribute to the development, but was perceived as patronising by the developers, who reported discomfort during interaction; the discussion was then extended to the overall communication between developers and users, which, especially over the forum, was not always smooth, sometimes leading to communication breakdowns in a literal sense:

Dev2: "There are different kinds of users: on the one hand some are really careful or even shy, but on the other hand some are kind of arrogant [. . .] I prefer not to answer myself, but wait for someone that replies in a better way than I would."

This opened a reflection on the role of students in the project and the perception of participation within the Smart Campus community. To further explore this kind of issues, the first interview study was organised. All members of the Smart Campus team reported being aware of the project aiming towards participatory development [42] and having a positive attitude towards the active participation of the users' community in the project. However, when probed at a deeper level, it became evident that the concept of involvement was intended more as informative, rather than participative [22]: instead of becoming true partners in the design process, students were expected to just express their opinions, which had to be taken into account by the lab. Especially in the case of developers, therefore, users were perceived as being requirement providers and application testers (i.e. informants), without any active role during the design and development stages:

Dev2: "External persons can suggest ideas, report bugs and the team has to listen to them and consider them for the next steps."

Dev2: "They are free to suggest ideas and even concrete improvements and we do take the information seriously, because they will use the application and that's why their opinion is important."

A designer indeed explicitly raised a concern about the lab understanding of participation:

Des4: "Sometimes I think that we don't listen very much to the students. We need feedback, we have to make more effort to understand and to listen to their opinions [. . .]. Sometimes we take decisions without a real participatory design approach. We take the decisions from the top of the hierarchical organization. This is only for making it easier and faster. We cannot listen to every little thing from the students."

We can see from these quotations that the perception of the community role retained some UCD/PD elements in designers and some Agile elements in developers: for instance, developers maintained an understanding of the customer as the funding agent whose requirements, although changing, were binding; on the other hand, designers expressed unease at their limited possibility to fully take into account the needs coming from the community. Yet, the compromise resulting from the integration of the two methodologies was unclear. In fact, the community was not understood as a proper customer, as it was not expected to actively participate in the management of the development process (for instance, it was not part of the community's duties to prioritise needs) and it appeared unable to adequately articulate its requirements for developers. On the other hand, the community lost part of its prerogatives as a user, since the management ultimately decided how to steer the direction of the project.

Indeed, students themselves were aware of the importance of their involvement, but they still confined it to a role of informants and consultants rather than of active participants who could effectively modify the outcome of the project:

Stud2: "The impact is strong: most of the participants are checking [Smart Campus] out and using at least one app frequently."

Stud10: "As a tester, you give me the smartphone and I can give you feedback. I can do something in exchange".

This attitude was evident also in the forum, which counted approximately 500 users who wrote about 2000 posts: over 67 % of threads reported problems and issues with the applications or the smartphone, and only 27 % reported suggestions for the evolution of the project. This in turn raised different expectations about what kind of contribution users could provide and what kind of feedback needed to be returned to them, in a vicious circle: being users' posts seldom focused on proposals for improvement, developers increasingly consolidated a perception of the community as a group of testers, therefore not entirely committing to acknowledging the actual suggestions for functionality enhancement; as a result, such kind of contributions seemed to appear less over time.

6.4.2 Documentation

As exemplified by the question asked by *Dev1: "What do you mean by documenting?"*, documentation did not appear to have an intrinsic value as a communication tool, neither for developers nor for interns; the first ones wrote it when explicitly required, the second ones mainly because they had to report to supervisors.

Dev3: "I document some piece of code [...] only if somebody asks me to because it is needed by others. Otherwise it's rare that a developer comments his code."
Dev4: "I document my development process sometimes, because it depends on the time that I have... If I have time, I spend some time to write a document"

To further investigate this aspect, the second interview study was organised. Since the development team was collocated and consisted of a limited number of members, writing documentation became just overhead in practice, as developers found it arguably quicker and easier to just meet and discuss in person within the office:

Dev7: "For the discussion between developers with different points of view, it's quicker to go to the office with the other developer and discuss it."

One of the interns however explicitly realised that this tended to create a closed, connected, fast-paced group:

Int3: "In my opinion, developers should participate in the forum, that is talk to the users without staying in a closed group. This would be counter-productive, as we [developers] meet every day."

The limited actual documentation was typically written and located within the code in the form of comments; developers shared more extensive online documents in case they needed to describe a complex process or provide more detailed information on what they did to a colleague who was meant to take over their work. Graphical documentation (wireframes and quick hand-drawn sketches) was used to discuss interface design, to align work between designers and developers and to supplement the written one in illustrating complex processes; it was usually transient and kept for personal use.

Dev5: "Well, the code is open source, so . . . they read my code and in the middle of my code there is some comment"
Dev5: "If it's a UI feature, I try to draw it . . . But I do not share this kind of sketches with anyone. It's just for me . . . when I complete the feature, I throw them away."

The attitude with regards to documentation instead changed in the case of designers or of interns who did not typically share the same space and time in the lab anymore due to academic commitments or end of their internship: they tended to habitually use documentation to organise and manage their progress. Designers regularly shared reports and reflections through Dropbox and online documents; interns reported first using everyday objects such as post-its and then moving to shared online documents as well as their collocation became less frequent.

Int5: "At first we used a lot of post-its that we would stick on the whiteboard and we work there as a group . . . we wrote all the points to develop, what one would do, what the other would do . . . when I basically remained the only one working there, I started to use shared online documents to communicate with other developers . . . and then that became the main communication method . . . The same things that we would write on post-its, we would then write on these shared documents."

The introduction in the team of several members (typically students) who worked during different shifts, remotely, or from different locations however reduced the chances of non-mediated communication while increasing the need for a shared, accessible knowledge base: yet, how to effectively support such need while leveraging on existing working practices remained an open point.

In general, the interviews showed how a series of attempts at effectively supporting documentation yielded contrasting results. In the following paragraphs we summarise what happened with reference to different articulation platforms used to facilitate the communication within the project team (developers, designers, and interns) and between the team and the user community.

Developers' Wiki This platform mainly hosted technical documentation for internal use. Developers reported checking it almost exclusively to read information; active contributions had been generally limited to the first steps of the project, as the lack of guidelines on how to structure the wiki rapidly led to a very confusing articulation which finally resulted in the CTO of the project being the only one "legitimated" to write content in it. Developers stated that they seldom edited minor points, typically to update sections concerning the tasks they were working on:

Dev1: "I know that on the Smart Campus Developers' wiki every library has documentation . . . these are done by G. and R. [CTO]"

GitHub Both interns and developers recognised in principle the relevance of documentation for involving external developers from the community and promoting the project, and some acknowledged that it should be placed in GitHub along with the code; in practice, however, developers did not use GitHub for documentation beyond the comments attached to code commits.

Dev4: "If I use open software, it's more important to publish the code and the documentation because if a developer needs to use my code he can learn where it is, how it works and so."

In fact, GitHub was considered to be quite cumbersome to search and likely to be too "geeky" to be widely used by the users' community.

Dev3: "I find [GitHub] quite hard to navigate, because there is development, tips, hints … there should be some guidelines"

Forum While the wiki and GitHub were more oriented to facilitating communication within the Smart Campus team, this bulletin board was aimed to open a dialogue between the project team and the community of users and was perceived as more suitable for this purpose.

Int1: "If [the forum] is for documentation purposes within the developers and the community, it can make sense. If it is just between me and the other people of the group, I don't think so."

One of the students claimed that the forum could indeed be a good tool for supporting the open source project and discussions related to code:

Stud1: "It would be important [to have a dedicated section for developers] to help each other and exchange information and opinions about code and development."

Yet, while students perceived the forum as a place for discussion, developers saw it just as a unidirectional informative tool from the users to the lab and did not perceive it as a suitable support for shared project documentation. Because of this, and despite the forum being acknowledged as a rich source of information, it was not seen as a suitable support for documentation, but at most as a supplement.

Dev1: "[The forum] It's not an instrument designed for that … the documentation should be more structured documents where I can navigate like in a tree and search … It's something that is missing in the forum, but it's not designed for that"

Int4: "If there is the official documentation and you want to go deeper into a topic, then you can visit that section of the forum … "

Moreover, developers interpreted engaging in conversations on the forum as extra work that got a relevant priority only if it was related to their current tasks:

Dev2: "[I use the forum] only to receive feedback on the application situation."

Dev2: "[The forum] It's a good way to interact with people and understand … but to document the development of the topic, it's not good."

Interestingly, however, while interns generally acknowledged the relevance of the forum to interact with the community, they tended to check it less and less often as their role in the project became more similar to that of a proper developer.

Int3: "The forum is an additional thing, I do not get any notifications, I have to go check it myself."

6.4.3 Synchronisation of Iterations

In Smart Campus, where the customer was actually a whole community, and as it also happens in mobile app development more in general (let us think of the wealth of apps published on Apple's App Store or on Google Play), the amount of user feedback escalated quickly: in our opinion, this has been one of the most disruptive elements in an effective synchronisation of the periods of UCD and Agile. As a side note, we report that feedback management and prioritisation is often mentioned in the software engineering community (e.g. [51, 52]) as a major issue in Agile projects.

At least in the first stages of Smart Campus, some members of the management team were appointed to review user feedback and prioritise it before passing it on to designers; this contributed to complicating the attempt of designers to find a suitable balance with the development pace. As a result, the development team sometimes took design decisions on their own, proactively checking the forum and looking for suggestions to implement or bugs to fix without waiting for the designers or the management to filter the information for them, but rather relying on the users' community contribution; the design team then resorted to many ad-hoc design interventions performed through a face-to-face interaction with one of the developers requesting it.

D2: "To read the situation [. . .] of some application that I am developing in part [. . .] I read the application topic to understand if there are some problems"

Despite developers being generally sensitive to HCI themes, this situation often caused parts of the apps to be modified in subsequent iterations in order to better fit the holistic UX design, thus requiring the same piece of interface to be implemented over and over again, each time differently. Different studies prove that the synchronisation of UCD and Agile can indeed be complex: for instance, in [15], the length of iterations in the two approaches and their "different timescales" are identified as factors contributing to the difficulties in aligning designers and developers; however, the authors report development being significantly slower at prototyping than design, while we experienced the opposite situation.

In fact, what was discussed earlier about the perception of community involvement also influenced the struggle for effective feedback management, especially affecting how feedback was returned to users. Students in fact often asked for more transparency on the project organisation and for greater feedback about their suggestions; the developers however claimed that these aspects were ensured in the forum. Actually, suggestions provided on the forum were either addressed directly by developers or mediated by designers; in both cases, a ticket was created in an internal system and was then closed upon solution of the issue. However, users had no way to access these tickets or to know whether their proposals were being taken into account unless someone from the lab explicitly notified this again on the forum, which was infrequent. The situation was expected to improve with the introduction of GitHub, as the ticketing system became public, yet this approach remained quite obscure for users, as if the project team did not perceive the need to keep them informed about the outcome of their suggestions.

6.5 Discussion

The present work has discussed a project adopting both UCD and Agile in the context of mobile application development for a user community: reflecting on such experience has allowed identifying three main communication breakdowns that may hamper the fruitful integration of the two approaches if the user/customer is no longer uniquely identified, namely a variable interpretation of user involvement, a mismatch in the value of documentation, and a misalignment in iteration phases. We acknowledge that the data we presented are to be conceived within the specific context of our case study: yet, the three themes are also discussed in literature from different points of view, even though not systematically. Therefore, given the data collected from the project and the literature review that reinforced them, we propose some considerations on how to favour the convergence of UCD and Agile in a setting oriented towards a user community and to leverage on its benefits.

6.5.1 User Involvement

We have seen the difficulties in defining the role of the Smart Campus user community and how these affected several aspects of the project. Misunderstandings in user involvement have been echoed also in the literature: for instance, authors debate on the extent of user/customer involvement both in the UCD [5, 22] and in the Agile perspectives [15]. In the case of the integration of UCD and Agile for a user community, we propose to draw inspiration from participatory design [32] because of its intrinsic familiarity with the involvement of a variety of users, each representing its own needs and values, and with the resulting complexity. This was attempted also in Smart Campus by seeding a user community that could communicate directly with the developers through the forum and the beta testing experimentation. We believe that the availability of a platform such as the forum, supporting the dialogue not only between the community and the project team, but also within the members of the community, has been beneficial for the maturation of a sense of ownership of the users with respect to the applications and for a livelier discussion about the needs of the community itself.

We have also referred to our instantiation of participatory development [42], i.e. integrating users in the development team by having interns in the lab. This peculiar context has in our opinion given some advantages to the development process: by providing a direct, explicit link between the project team and the community, interns allowed mutual learning to take place and contributed a domain knowledge that resulted in a more informed feedback management; on the other hand, however, as found in [28, 32], such bond with the community became less and less meaningful as the time spent by students in the lab passed and they transformed into developers.

We acknowledge that the nature of Smart Campus as a funded R&D project put it in a privileged position that could afford having interns from a largely technically skilled user community as part of team. Yet, despite Smart Campus being a very specific case, we argue that, particularly when addressing a user community as it is often the case in mobile app development, informing the design and development process with participatory elements is still a valid suggestion and, retaining the richness and articulation of the voices of the community, is likely to be a sustainable approach in the long term.

This also provides a possible answer to the concerns appearing in literature about the customer's role in Agile: this figure is entitled to steer the direction of the project by redefining and re-prioritising his/her own requirements even while development is on-going [12], but at the same time he/she is overwhelmed by responsibility [25], his/her actual representativeness is questioned [3, 28, 31] and he/she is likely not to have the competence to exert such decision-making power, to the point that some authors have suggested that a member of the development team is most suited to play this role [29] instead. We believe that the context of mobile application development might open further possibilities if the community is composed of technically skilled people, as it was partly the case in Smart Campus: in this case, in fact, the decision-making power of users can be extended to cover choices that concern not only design, as it happens in participatory design, but also development.

6.5.2 Documentation

In our experience documentation can be kept to a minimum, as encouraged by the Agile principles [12, 33], and intended only for internal use as long as the development process is confined within a lab, as it was in our case. Yet, if we envision a scenario in which users are active participants engaged in participatory development, documentation becomes a means to ensure greater transparency over the development process, especially for coordinating the evolution of an open piece of software; the theme of geographical distribution is by the way also presented in [53] as one of the main differences between the open source and the Agile approaches. Furthermore, in [54] authors highlight that shifting to a distributed team, and thus having to create high-quality documentation and specifications, "requires different types of competences than simply expertise in programming and concomitant tacit knowledge": professional identities and work practices change, as the "articulation work" required to coordinate becomes a larger share of regular work.

6.5.3 Synchronisation of Iterations

We have also seen how the amount of user feedback coming from a community can quickly escalate and how handling it can affect the smoothness of the development process. Despite information loss in feedback management being unavoidable [52], we believe that an organisational culture informed by participatory design is likely to appropriately recognise the value in this continuous feedback and effectively handle it, integrating the users' voice while retaining as much as possible of its articulation. This context may also ensure greater transparency over the development process, showing in an organic way what the outcome of the feedback and suggestions provided by the community was and therefore establishing a dialogue with users, rather than having communication flowing just unidirectionally as it happened in the Smart Campus forum.

Indeed, how to effectively achieve this remained an open point in Smart Campus, as the project team was not able to envision a lightweight process for feedback implementation that did not interfere with the speed of the development pace; as a result, design lagged behind development, differently from what reported in [15] and suggested in [29, 40]. Yet, we believe that, as also shown by several suggestions in literature [e.g. 55], the management of user input can be facilitated if the integration of UCD and Agile fully occurs only after the conceptual design has been finalised, i.e. after the so-called "Iteration 0" [41], which can even be regarded as exceeding the scope of Agile methods [55]. In Smart Campus, for instance, while the user research was being performed, the backend functionalities of the platform were being developed as well. Once the conceptual design is ready, UCD can address the interface design, while Agile can proceed with the implementation of the business logic. In our opinion, it is likely that the feature-oriented framing of Agile is somehow too constraining in respect of the creativity and flexibility that characterise the early stages of UCD. A critical point is however still present in the handover of the conceptual design from this stage, where UCD and Agile proceed in parallel, to the subsequent iterations, where the two approaches merge.

6.5.4 Fostering Integration

We finally remark that as the integration between the UCD and Agile methodologies occurs, a compromise seems to be needed between their respective understandings of the user and the customer, both in terms of working practices [37] and in terms of organisational vision, especially in the case where the user/customer is no longer uniquely defined and is rather replaced by a community like in Smart Campus. In fact, some authors advocate the establishment of a suitable managerial and organisational context as one of the conditions for the integration of UCD and Agile to effectively happen [7, 37]. Clearly, an organisational culture that values participation and recognises the relevance of user input throughout the whole design and development process is likely to endorse a conceptual design that takes the users' voice into account, acknowledging and actively addressing the issues related to the responsibility towards users' needs and to the risk of losing track of the holistic UX design over time [7, 37].

In order to foster such a receptive organisational culture, we propose the adoption of design thinking [56], a methodology grounded on a "human-centred design ethos" that pervades all stages of a product lifecycle, from inspiration to ideation to implementation. This discipline leverages on "the designer's sensibility and methods to match people's needs with what is technologically feasible" and marketable, acknowledging the "value of a holistic design approach". In his work, Brown emphasises that design thinking is not just a prerogative of people in design schools, but is rather an attitude that can be assimilated also by other professionals. For what concerns specifically the integration of UCD and Agile, we believe that this perspective can foster an organisational culture which values and achieves empathy with users and endorses, both in the project management and its team, a common awareness of elements such as the relevance of user needs, a holistic UX vision, and a shared acceptance and ownership of the conceptual design and ultimately of the product.

6.6 Conclusions

In this paper we have proposed three themes that can be used as an analytical tool in the management and facilitation of projects involving UCD and Agile. Such themes, or communication breakdowns, concern potential mismatches in the formalisation of key concepts in the two approaches, namely the interpretation of user involvement, the value of documentation, and the synchronisation of iterations; they emerged from a case study in mobile application development for a user community and were reinforced with a literature review. We believe that reconciling them by promoting a receptive organisational culture that draws inspiration from participatory design and design thinking can be a fruitful way to effectively integrate UCD and Agile.

Acknowledgements Smart Campus was funded by TrentoRISE. The analytical work presented in this chapter has been possible thanks to the funding granted by the Italian Ministry of Education, University and Research (MIUR) through the project "Città Educante", project code CTN01_00034_393801. We wish to thank the Smart Campus team and all the students who contributed to the project.

References

1. Salah D, Paige RF, Cairns P (2014) A systematic literature review for agile development processes and user centred design integration. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering. ACM, New York, p 5
2. Kane D (2003) Finding a place for discount usability engineering in agile development: throwing down the gauntlet. In: Agile development conference, 2003. ADC 2003. Proceedings of the IEEE, pp 40–46
3. Sohaib O, Khan K (2010) Integrating usability engineering and agile software development: a literature review. In: Computer design and applications (ICCDA), 2010 International conference on, vol 2. IEEE, pp V2–V32)
4. Jurca G, Hellmann TD, Maurer F (2014) Integrating Agile and user-centered design: a systematic mapping and review of evaluation and validation studies of Agile-UX. In: Agile conference (AGILE), 2014. IEEE. pp 24–32
5. Damodaran L (1996) User involvement in the systems design process-a practical guide for users. Behav Inform Technol 15(6):363–377
6. Kujala S (2003) User involvement: a review of the benefits and challenges. Behav Inform Technol 22(1):1–16
7. Cajander Å, Larusdottir M, Gulliksen J (2013) Existing but not explicit-the user perspective in Scrum projects in practice. In: Human-computer interaction–INTERACT 2013. Springer, Berlin/Heidelberg, pp 762–779
8. Frayling C (1993) Research in art and design. Royal College of Art, London
9. Bjørn P, Ngwenyama O (2009) Virtual team collaboration: building shared meaning, resolving breakdowns and creating translucence. Inf Syst J 19(3):227–253
10. Rogers Y, Sharp H, Preece J (2011) Interaction design: beyond human-computer interaction. Wiley, Chichester
11. Norman DA (1986) Cognitive engineering. In: User centered system design. L. Erlbaum Associates, Hillsdale, pp 31–61
12. Beck K, et al. Manifesto for Agile software development. http://www.Agilemanifesto.org
13. Verdiesen B (2014) Agile user experience. MSc dissertation, Radboud University Nijmegen, Nijmegen
14. Clark HH, Brennan SE (1991) Grounding in communication. In: Perspectives on socially shared cognition, vol 13. American Psychological Association, Washington, DC, pp 127–149
15. 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. Springer, Berlin/Heidelberg, pp 143–153
16. Wolkerstorfer P et al (2008) Probing an Agile usability process. In: Proceedings of CHI 2008. ACM Press, New York, pp 2151–2157
17. McGinn J, Chang AR (2013) RITE+Krug: a combination of usability test methods for Agile design. J Usability Stud 8(3):61–68
18. Memmel T, Gundelsweiler F, Reiterer H (2007) Agile human-centered software engineering. In: Proceedings of the 21st British HCI group annual conference on people and computers: HCI... but not as we know it-Volume 1. British Computer Society, Swinton, pp 167–175
19. Ungar JM, White JA (2008) Agile user centered design: enter the design studio – a case study. In: Proceedings of CHI 2008. ACM Press, New York, pp 2167–2177
20. Gothelf J (2013) Lean UX: applying Lean principles to improve user experience. O'Reilly Media, Inc., Beijing
21. Liikkanen LA, Kilpiö H, Svan L, Hiltunen M (2014) Lean UX: the next generation of user-centered agile development? In: Proceedings of the 8th Nordic conference on human-computer interaction: fun, fast, foundational. ACM, New York, pp 1095–1100
22. Iivari J, Iivari N (2011) Varieties of user-centredness: an analysis of four systems development methods. Inf Syst J 21(2):125–153

23. Bjerknes G, Bratteteig T (1995) User participation and democracy: a discussion of Scandinavian research on system development. Scand J Inf Syst 7(1):1
24. Ambler SW. Introduction to user stories. http://www.agilemodeling.com/artifacts/userStory.htm. Accessed 11 Dec 2015
25. Martin A, Biddle R, Noble J (2004) The XP customer role in practice: three studies. In: Proceedings of the ADC 2004. IEEE. pp 42–54
26. Schwartz L (2014) Agile-user experience design: does the involvement of usability experts improve the software quality? Int J Adv Softw 7(3&4):456–468
27. Sharp H, Robinson H (2004) Integrating user-centred design and software engineering: a role for extreme programming? http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.4554&rep=rep1&type=pdf
28. Beyer H, Holtzblatt K, Baker L (2004) An Agile customer-centered method: rapid contextual design. In: Extreme programming and Agile methods-XP/Agile universe 2004. Springer, Berlin/Heidelberg, pp 50–59
29. Sy D (2007) Adapting usability investigations for Agile user-centered design. J Usability Stud 2(3):112–132
30. Schwaber K, Sutherland J (2011) The Scrum guide. Scrum.org
31. Schwartz L (2013) Agile-user experience design: with or without a usability expert in the team? In Proceedings of the ICSEA 2013. IARIA, pp 359–363
32. Gregory J (2003) Scandinavian approaches to participatory design. Int J Eng Educ 19(1): 62–74
33. McInerney P, Maurer F (2005) UCD in Agile projects: dream team or odd couple? Interactions 12:19–23
34. Brown JM, Lindgaard G, Biddle R (2011) Collaborative events and shared artefacts: Agile interaction designers and developers working toward common aims. In: Agile conference (AGILE), 2011. IEEE, pp 87–96
35. Selic B (2009) Agile documentation, anyone? Softw, IEEE 26(6):11–12
36. Ambler SW. Agile/Lean documentation: strategies for agile software development. http://www.agilemodeling.com/essays/agileDocumentation.htm. Accessed 6 Nov 2015
37. Lárusdóttir MK, Cajander Å, Gulliksen J (2012) The big picture of UX is missing in Scrum projects. In: Proceedings of the 2nd international workshop on the interplay between user experience evaluation and software development, in conjunction with the 7th Nordic conference on human-computer interaction. http://ceur-ws.org/Vol-922/I-UxSED-2012-Proceedings.pdf#page=49. Accessed on 11 Dec 2015
38. Gould JD, Lewis C (1985) Designing for usability: key principles and what designers think. Commun ACM 28(3):300–311
39. Miller L (2005) Case study of customer input for a successful product. In: Proceedings of Agile. pp 225–234
40. Nodder C, Nielsen J (2010) Agile usability: best practices for user experience on Agile development projects. Nielsen Norman Group, Fremont
41. Fox D, Sillito J, Maurer F (2008) Agile methods and user-centered design: how these two methodologies are being successfully integrated in industry. In: Agile, 2008. AGILE'08. Conference. IEEE, pp 63–72
42. De Angeli A, Bordin S, Menéndez Blanco M (2014) Infrastructuring participatory development in information technology. In: Proceedings of the 13th participatory design conference: research papers, vol 1. ACM, pp 11–20
43. De Angeli A, Bordin S, Menéndez Blanco M (2014) Reflections over a socio-technical infrastructuring effort. In: Proceedings of 2nd Workshop on Cultures of Participation in the Digital Age, CoPDA'14, Como, Italy, May 27, 2014, CEUR-WS.org, online CEUR-WS.org/Vol-640/paper1.pdf
44. Menéndez Blanco M, Bordin S, De Angeli A (2014) Sociotechnical infrastructuring for participation. Workshop on cooperative technologies in democratic processes – Beyond e-Voting, COOP. http://www.iisi.de/fileadmin/IISI/upload/IRSI/2014Vol11Iss1/IRSI_Vol11_Iss1_Menendez_Bordin_De_Angeli_Socio-technical_infrastructuring_for_participation.pdf. Accessed on 11 Dec 2015

45. Bordin S, Menéndez Blanco M, De Angeli A (2014) ViaggiaTrento: an application for collaborative sustainable mobility. EAI Endorsed Trans Ambient Sys 14:4
46. Bordin S, Menéndez Blanco M, De Angeli A (2014) Catch me if you can: reconciling Agile and UCD. https://ucdandagile.files.wordpress.com/2014/10/nr2-nordichi2014-agile-ucd-workshop-bordin.pdf, Workshop on the Integration of UCD and Agile Development, NordiCHI 2014
47. Teli M, Bordin S, Blanco MM, Orabona G, De Angeli A (2015) Public design of digital commons in urban places: a case study. Int J Hum Comput Stud 81:17–30
48. Sy D, Mille L (2008) Optimizing Agile user-centred design. In: CHI'08 extended abstracts on Human factors in computing systems. ACM, pp 3897–3900
49. Neustaedter C, Sengers P (2012) Autobiographical design in HCI research: designing and learning through use-it-yourself. In: Proceedings of the designing interactive systems conference. ACM, pp 514–523
50. Smith CP (1992) Motivation and personality: handbook of thematic content analysis. Cambridge University Press, Cambridge, MA
51. Gartner S, Schneider K (2012) A method for prioritizing end-user feedback for requirements engineering. In: Cooperative and Human Aspects of Software Engineering (CHASE), 2012 5th international workshop on, IEEE. pp 47–49
52. Lee MJ, Ko AJ (2012) Representations of user feedback in an Agile, collocated software team. In: Cooperative and Human Aspects of Software Engineering (CHASE), 2012 5th international workshop on, IEEE. pp 76–82
53. Goldman R, Gabriel RP (2005) Innovation happens elsewhere: open source as business strategy. Morgan Kaufmann, Amsterdam
54. Matthiesen S, Bjørn P, Petersen LM (2014) Figure out how to code with the hands of others: recognizing cultural blind spots in global software development. In: Proceedings of the CSCW 2014. ACM press, pp 1107–1119
55. Schwartz L (2013) Agile-user experience design: an Agile and user-centered process?. In: ICSEA 2013, The Eighth International Conference on Software Engineering Advances. IARIA XPS Press
56. Brown T (2008) Design thinking. Harv Bus Rev 86(6):84