UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE

ICT International Doctoral School

A FRAMEWORK FOR INTEGRATING USER-CENTRED DESIGN AND AGILE DEVELOPMENT IN SMALL COMPANIES

Silvia Bordin

Advisor: Prof. Antonella De Angeli, Università degli Studi di Trento

Reviewers: Prof. Helen Sharp, Open University;

Dr. Peggy Gregory, University of Central Lancashire

Defence committee: Prof. Fabio Casati, Università degli Studi di Trento

Prof. Vincenzo D'Andrea, Università degli Studi di Trento

Dr. Peggy Gregory, University of Central Lancashire

Abstract

The integration of user-centred design (UCD) and Agile development is gaining increasing momentum in industry: the two approaches show promising complementarities and their convergence can lead to a more holistic software engineering approach relative to the application of just one of them. However, the practicalities of this integration are not trivial and the topic is currently of interest to a variety of research communities. This thesis aims at understanding the integration of user-centred design and Agile development and at supporting its adoption in small companies. Based on a qualitative approach, it is positioned at the intersection of Computer Supported Cooperative Work and software engineering, and is grounded on several empirical studies performed in industry.

The work is organised in three stages inspired by the action research approach. The first stage was dedicated to understanding software development practice: through literature review and two ethnographically-informed studies, it resulted in the first contribution of this thesis, that is a set of communication breakdowns that may hinder the integration of user-centred design and Agile development.

The second stage was dedicated to deliberating improvements of practice: the set of communication breakdowns was elaborated into the second contribution of this thesis, that is a framework of focal points meant to help the organisation diagnose and assess communication breakdowns in its work practice.

The third stage of the thesis was dedicated to implementing and evaluating improvements. The framework was further elaborated into a training on the adoption of the framework itself. Such training was instantiated in two iterations of action research performed in small development organisations, with the aim of establishing a supportive organisational environment and mitigating communication breakdowns. Once validated through these cases, the training constituted the third contribution of this thesis. Results show that the intervention has benefited companies at several levels, enriching work practice with fresh techniques, favouring team collaboration and cooperation, and resulting in a shift from a technology-centred mindset to a more user-centred one.

Keywords

Qualitative research; empirical software engineering; organisational context; software development practice.

Statement of contribution

This thesis reports research principally done by the author, as a part of her doctoral studies. The work presented in Chapters 4, 5, 7, and 8 has been published as follows. Parts of these papers have been re-interpreted and rewritten in the dissertation; some passages have been quoted verbatim.

The study in Chapter 4 was published as: Bordin, S., & De Angeli, A. (2016). Communication breakdowns in the integration of user-centred design and Agile development. In *Integrating User-Centred Design in Agile Development* (pp. 137-161). Springer International Publishing.

In particular, Section 4.1 was published in parts as:

De Angeli, A., Bordin, S., & Menendez Blanco, M. (2014). Infrastructuring participatory development in information technology. In *Proceedings of the 13th Participatory Design Conference: Research Papers-Volume 1* (pp. 11-20). ACM.;

Bordin, S., Menendez Blanco, M., & De Angeli, A. (2014). ViaggiaTrento: an application for collaborative sustainable mobility. *ICST Trans. Ambient Systems*, 1(4), e5;

Teli, M., Bordin, S., Blanco, M. M., Orabona, G., & De Angeli, A. (2015). Public design of digital commons in urban places: a case study. *International Journal of Human-Computer Studies*, 81, 17-30.

The study in Chapter 5 was published as Bordin, S., & De Angeli, A. (2016). Focal points for a more user-centred Agile development. In *International Conference on Agile Software Development* (pp. 3-15). Springer International Publishing.

Parts of Chapter 6 and the study in Chapter 7 are currently under review as Bordin, S., & De Angeli, A. A focal points framework for supporting the integration of UCD and Agile in practice. Submitted to *Computer-Supported Cooperative Work*.

The study in Chapter 8 was partly published as Bordin, S., & De Angeli, A. (2017). Inoculating an Agile Company with User-Centred Design: An Empirical Study. In *International Conference on Agile Software Development* (pp. 235-242). Springer, Cham.

Acknowledgements

This PhD has offered me a lot of opportunities: it allowed me to discover where my talent is, learn what I am capable of, and acknowledge where my limits are. First of all, I would like to thank my advisor, prof. Antonella De Angeli, for the giving me the chance to begin and pursue this journey.

Then, I would like to thank the committee for kindly reviewing my dissertation, providing useful insights, and taking the quality of my work to a much higher level. In addition, I would like to thank the professors and researchers I met during the months I spent in the United Kingdom, for opening my mind and supporting me beyond expectations. Many thanks also to EIT Digital for having funded this experience, thus making it possible.

I also wish to thank my mom and dad for supporting me during this journey. This year has brought many new challenges for the whole family, and I wish them the strength to address these challenges and find a new balance. Many thanks to my fellow PhD candidates, for sharing the hardships and celebrating publications, and to my girlfriends, for reminding me there is a world outside academia. Finally, I wish to thank Andrea for lovingly feeding me, for helping me put things in perspective, and for being boldly confident in my capabilities.

Table of contents

ABSTRACT 3

STATEMENT OF CONTRIBUTION 5

ACKNOWLEDGEMENTS 6

TABLE OF CONTENTS 7

1 INTRODUCTION 12

1.1 MOTIVATION 12

1.2 RESEARCH QUESTIONS AND CONTRIBUTIONS 13

1.3 DISSERTATION OUTLINE 13

2 STATE OF THE ART 15

2.1 FOUNDATIONAL CONCEPTS 15

2.1.1 USER-CENTRED DESIGN 16

2.1.2 AGILE DEVELOPMENT 17

2.2 A QUALITATIVE RESEARCH APPROACH TO SOFTWARE ENGINEERING 18

2.3 INTEGRATION OF USER-CENTRED DESIGN AND AGILE DEVELOPMENT 21

2.3.1 STATE OF THE ART IN THE INTEGRATION 22

2.3.2 COMMUNICATION BREAKDOWNS 25

2.3.2.1 User and customer 27

2.3.2.2 Role of documentation 28

2.3.2.3 Synchronisation of iterations 29

2.3.3 CONCLUSION 30

3 METHODOLOGY 31

3.1 ABOUT QUALITATIVE RESEARCH 31

3.2 RESEARCH DESIGN 32

3.3 UNDERSTANDING PRACTICE: ETHNOGRAPHICALLY-INFORMED STUDIES 34

3.4 IMPROVING PRACTICE: ACTION RESEARCH 35

3.4.1 ACTION RESEARCH 35

3.4.2 COOPERATIVE METHOD DEVELOPMENT 38

3.5 DATA COLLECTION AND ANALYSIS 39

3.5.1 SEMI-STRUCTURED INTERVIEWS 39

3.5.2 THEMATIC ANALYSIS 40

4 STUDY 1: THE SMART CAMPUS PROJECT 42

4.1 THE CONTEXT 42

4.1.1 PROJECT METHODOLOGY 44

4.1.2 DATA COLLECTION AND ANALYSIS 46

4.2 RESULTS 47

4.2.1 USER VS. CUSTOMER 47

4.2.2 ROLE OF DOCUMENTATION 50

4.2.3 SYNCHRONISATION OF DESIGN AND DEVELOPMENT 53

4.3 DISCUSSION 55

4.3.1 USER VS. CUSTOMER 56

4.3.2 ROLE OF DOCUMENTATION 57

4.3.3 SYNCHRONISATION OF DESIGN AND DEVELOPMENT 57

4.4 CONCLUSION 58

5 STUDY 2: H-UMUS 60

5.1 THE CONTEXT 60

5.1.1 DATA COLLECTION AND ANALYSIS 61

5.1.2 UNDERSTANDING PRACTICE 62

5.2 RESULTS 64

5.2.1 USER VS. CUSTOMER 64

5.2.2 ROLE OF DOCUMENTATION 65

5.2.3 SYNCHRONISATION OF DESIGN AND DEVELOPMENT 66

5.3 COMPARISON WITH SMART CAMPUS 67

5.4 CONSOLIDATING COMMUNICATION BREAKDOWNS 69

6 A DIAGNOSTIC FRAMEWORK 71

6.1 STATE OF THE ART 71

6.1.1 THE NEED FOR A RECEPTIVE ORGANISATIONAL ENVIRONMENT 72

6.1.2 TRAINING DEVELOPERS ON UX 74

6.2 FROM COMMUNICATION BREAKDOWNS TO FOCAL POINTS 76

6.3 APPLYING THE FRAMEWORK IN PRACTICE 81

7 STUDY 3: DELTA INFORMATICA 84

7.1 THE CONTEXT 84

7.2 UNDERSTANDING PRACTICE 85

7.2.1 PRELIMINARY INTERVIEW STUDY 86

7.2.2 PRELIMINARY WORKSHOPS 88

7.3 IMPLEMENTING IMPROVEMENTS: DESIGN WORKSHOPS 89

7.3.1 WORKSHOP 1 – PRODUCT OWNER IDENTIFICATION 91

7.3.2 WORKSHOP 2 – DESIGN SPACE 92

7.3.3 WORKSHOP 3 – NON FUNCTIONAL REQUIREMENTS 96

7.3.4 WORKSHOP 4 – TERMINOLOGY 96

7.3.5 WORKSHOP 5 – DESIGN COMPLETION 99

7.4 IMPLEMENTING IMPROVEMENTS: DEVELOPMENT WORKSHOPS 100

7.4.1 MEETING 1 – BACKLOG CREATION 100

7.4.2 MEETING 2 – SCRUM BOARD INTRODUCTION 101

7.4.3 MEETINGS 3 AND 4 – SPRINT REVIEWS 103

7.5 EVALUATION OF IMPROVEMENT 103

7.5.1 PROBLEM SOLVING INTEREST: USER EVALUATION 104

7.5.2 RESEARCH INTEREST: INTERVIEWS 105

7.5.2.1 Outcomes of deliberated improvements 105

7.5.2.2 About proposed techniques 108

7.5.2.3 Potential for improvement 110

7.6 DISCUSSION 113

8 STUDY 4: EMAZE 115

8.1 THE CONTEXT 115

8.2 UNDERSTANDING PRACTICE 116

8.3 IMPLEMENTING IMPROVEMENTS 117

8.3.1 WORKSHOP 1 – USER RESEARCH 119

8.3.2 WORKSHOP 2 - CONCEPTUAL DESIGN 121

8.3.3 WORKSHOP 3 - PROTOTYPING 122

8.3.4 WORKSHOP 4 - EVALUATION 125

8.4 EVALUATING IMPROVEMENTS 125

8.5 COMPARISON WITH DELTA 127

8.6 TRAINING ON THE FRAMEWORK ADOPTION 129

9 DISCUSSION 131

9.1 LIMITATIONS 134

9.2 FUTURE WORK 135

10 APPENDIX 137

10.1 SMART CAMPUS 137

10.1.1 FIRST INTERVIEW STUDY – PERCEPTION OF USER INVOLVEMENT 137

10.1.1.1 Contents 137

10.1.1.2 Interview guide 138

10.1.2 SECOND INTERVIEW STUDY – PROJECT DOCUMENTATION 142

10.2 H-UMUS 142

10.3 DELTA INFORMATICA 144

10.3.1 PRELIMINARY INTERVIEWS 144

10.3.2 EVALUATION INTERVIEWS 145

10.4 EMAZE 150

10.4.1 PRELIMINARY INTERVIEWS 150

10.4.2 EVALUATION INTERVIEW 151

BIBLIOGRAPHY 159

1 Introduction

This thesis aims at understanding the integration of Agile development and user-centred design and at supporting its adoption in small enterprises. This chapter motivates the relevance of this research problem, defines research questions and contributions of the thesis, and outlines the structure of the dissertation.

1.1 Motivation

In recent years there has been a growing interest in understanding the combination of user-centred design (UCD) and Agile development, both in the human-computer interaction (HCI) community and in the software engineering (SE) one. This convergence would lead to a more holistic software engineering approach relative to the application of one of the individual methodologies alone (Salah et al., 2014): it would combine the known positive aspects of Agile, such as transparency, inspection, and adaptation, with the possibility to be truly user-centred (Øvad and Larsen, 2016a). Reported benefits include improvements in communication within the team, in visibility over project goals, and in product usability (Da Silva et al., 2015).

The advantage of integrating the two approaches is two-fold: 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 (Kane, 2003; Sohaib and Khan, 2010). Yet, these aspects cannot be overlooked anymore, and a carefully designed UX can provide an advantage over competing products (Jurca et al., 2014) in terms of product acceptance and customer satisfaction (Damodaran, 1996; Kujala, 2003). In addition, the relevant costs of an extensive UCD process (Cataldo and Herbsleb, 2008) are counterbalanced by a reduction in the costs for user training and support (due to an improved usability (Ardito et al., 2014)) and for implementation, since unnecessary features or critical usability issues are spotted early in the development process (Nielsen, 1994).

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” (Salah et al., 2014) 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. 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.

1.2 Research questions and contributions

Although beneficial, the convergence of user-centred design (UCD) and Agile development is not trivial to achieve. With the aim of understanding how the integration of the two approaches occurs in practice and how to support it, this thesis unfolded over three stages: understanding practice, deliberating improvements, and implementing and evaluating improvements. The following research questions were addressed:

- **RQ1**: in which aspects of work practice do difficulties in integrating UCD and Agile appear?
- **RQ2**: what practical solution can be devised to address the difficulties identified in RQ1?
- **RQ3**: how can the solutions identified in RQ2 be adapted to the specific setting of small organisations?

Each research question has been answered by one contribution of this thesis respectively, as follows:

- **C1**: a set of potential communication breakdowns representing critical issues in the integration of UCD and Agile;
- **C2**: a framework of focal points supporting the organisation in shaping the integration of UCD and Agile;
- **C3**: a training approach on the adoption of the framework.

1.3 Dissertation outline

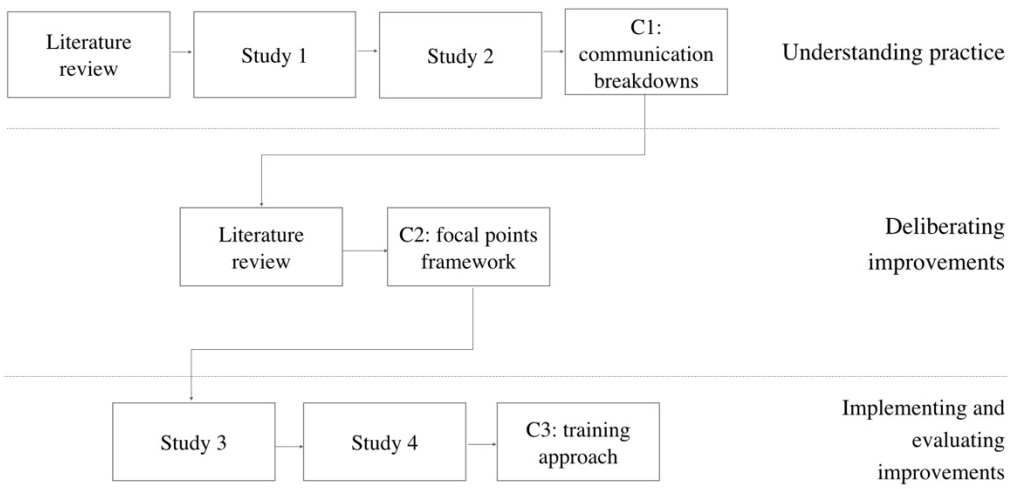


Figure 1 - dissertation outline

Figure 1 outlines the structure of the research described in this dissertation: as mentioned in section 1.2, it is divided in three stages, inspired by the Cooperative Method Development approach (Dittrich et al., 2008), an instantiation of action research: understanding practice, deliberating improvements, and implementing and evaluating improvements. The different research methods used in the studies that contribute to each stage are described in chapter 3.

The understanding practice stage began with an analysis of the state of the art on the integration of UCD and Agile, discussed in chapter 2. Such analysis positions the work within qualitative research in software engineering, reviews extant literature on attempts at integrating the two approaches, and extracts from it a set of sensitising concepts representing critical issues in the integration itself. These sensitising concepts then guided the interpretation of two ethnographically-informed studies, described in chapters 4 and 5 respectively. As a result, they were refined into contribution C1 as a set of communication breakdowns.

The deliberating improvements stage is described in chapter 6. Building on an additional literature review, this stage abstracted communication breakdowns into contribution C2, which consists of a framework highlighting key points on which the organisation should take decisions in order to shape the integration of UCD and Agile.

The implementing and evaluating improvements stage aimed at translating the framework back into practice: to this end, it instantiated a training on the adoption of the framework in two iterations of action research, described in chapters 7 and 8 respectively. The training represents contribution C3. Chapter 9 elaborates on the lessons learned throughout the research process, summarises the findings of this thesis, and points out directions for future work. Finally, the appendix in chapter 10 collects support materials for the studies presented.

2 State of the art

This chapter starts by defining the foundational themes of UCD and Agile, and then positions the work presented in this thesis with respect to qualitative research on software engineering, particularly relating to the Computer Supported Cooperative Work (CSCW) domain. The chapter then moves to a detailed discussion of attempts at integrating the two approaches reported in literature. This analysis allowed to identify three critical aspects that may negatively affect such integration.

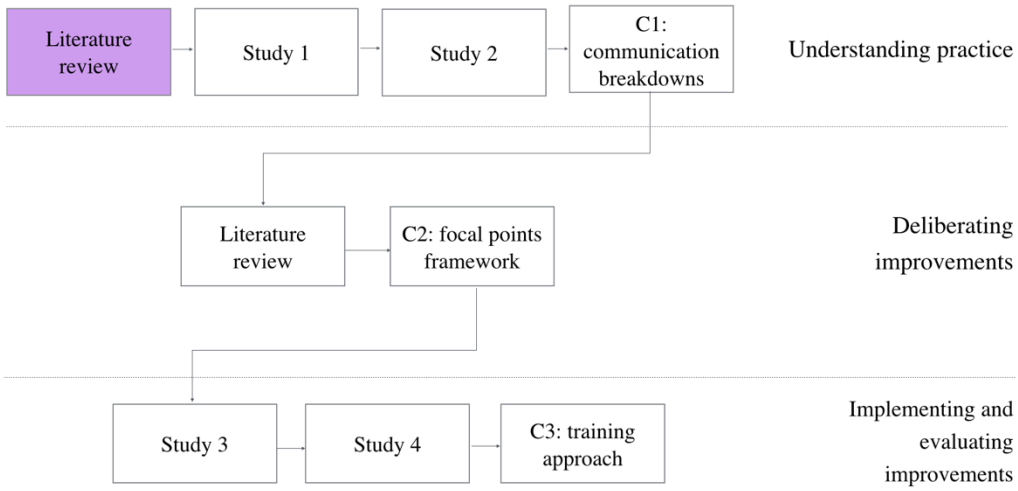


Figure 2 - dissertation progress: literature review

2.1 Foundational concepts

This work relies heavily on the practice paradigm (Kuutti and Bannon, 2014). Kuutti and Bannon describe it as examining long-term, persisting actions that are situated in time and space and depend on a variety of features of the surrounding environment, requiring coordination with others. The current situation is therefore seen as the momentary result of a historical evolution, which is constantly under the influence of a variety of forces. In this paradigm, the unit of intervention is the whole practice, intended not only as technology, but as everything related and interwoven in the performance; all of the aspects of the practice are under scrutiny and potentially changeable, based on the goals of the intervention. Because of this, research methods adopted in the practice paradigm have often been qualitative, *in situ*, and involving people, artefacts, organisational routines and daily practices.

Starting from within the practice paradigm, this thesis focuses on the integration of UCD and Agile approaches. The following paragraphs will provide the fundamental notions about these approaches, on which the rest of the dissertation will rely: the aim of this thesis is in fact to understand the integration of Agile development and user-centred design and at supporting its adoption in small enterprises.

2.1.1 User-centred design

User-centred design (UCD) 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 (Rogers et al., 2011). In this perspective, "the purpose of the system is to serve the user, not to use a specific technology, not to be an elegant piece of programming" (Norman, 1986, p. 61); because of this, some authors consider UCD "the key to product usefulness and usability" (Mao et al., 2005, p. 105). Serving the user implies that the product must exhibit a good usability: this can be achieved by eliciting accurate and frequent user feedback (Rogers et al., 2011), thereby refining the design in an iterative fashion (Ferreira, 2008). The basic cycle of UCD in fact moves from an analysis of user needs to the design of potential solutions, which are then prototyped and evaluated again with the user; the results of such evaluation are then fed back to a new step of analysis and so forth (Rogers et al., 2011).

The concept of user-centredness has a variety of meanings, depending on the community and the period of reference. The literature review in (Iivari and Iivari, 2011) resulted in four dimensions of user-centredness: user focus, work-centredness, user involvement, and personalisation. In this dissertation, I will refer to user-centredness as user involvement, which authors understand as a broad concept encompassing the involvement of users in the system development process. This definition, borrowed from the Information Systems community, supersedes that of user-centredness as user participation, which refers specifically to an active engagement of users or their representatives in the process: in fact, user involvement can also refer to passive activities such as being an object of observation. Such interpretation is particularly relevant in a product development context, in which identifying and contacting prospective users for the purposes of user participation can be challenging.

In (Abelein et al., 2013), user involvement in the system development process is defined as "crucial", because "users of software […] are familiar with both the work and the context that the software system should support, thus it's critical to involve them in the process" (p. 17). Involving users in systems design results in high reward and several widely acknowledged benefits (Ardito et al., 2014; Hussain et al., 2009a; Kujala, 2003; Mao et al., 2005): these include "effectiveness, efficiency, satisfaction, wellbeing at work, brand loyalty, health and

safety, employee retention, respect for human dignity, and competitiveness" (Cockton et al., 2016, p. 2), improved quality, understanding, and acceptance of the product (Damodaran, 1996), and generally positive effects on both system success and user satisfaction (Abelein et al., 2013; Kujala, 2003).

2.1.2 Agile development

The overview in (Laanti et al., 2013) highlights that there are several definitions of Agile development and agility, each emphasising different features of the approach. Summarising the most common aspects of these definitions, the label "Agile development" can be referred to 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, informal meetings rather than through formal documentation. In addition, great emphasis is put on constant feedback from the customer, who is in fact expected to be actively involved in the development process, and to have the power to steer the direction of the project by intervening on the requirements according to his/her possibly evolving needs (Verdiesen, 2014). In the following, I will refer to Agile as the union of a variety of practices with a mindset with which to approach the problems at hand. It should in fact be noted that there is a widespread tendency to move Agile beyond the boundaries of software development, and rather looking at the agility of the whole business, as witnessed for instance in the work of the Agile Business Consortium (https://www.agilebusiness.org/) or in the recent rise of the Italian Agile Business Day events (http://agilebusinessday.com/). This school of thought is grounded on the understanding of several practitioners and researchers that *doing* Agile is not *being* Agile, as summarised for instance in (Lapham et al., 2011), and that "adopting Agile without the corresponding shift in consciousness is mostly pointless" (Provaglio, 2017).

Agile approaches are grounded on the principles and values listed in the Agile Manifesto (Beck, 2001), which emerged from practitioners as a response to the inadequacies they perceived in traditional or waterfall approaches. These are characterised by sequential activities and substantial documentation, and in turn by rigidity towards change (Robey et al., 2001): as the complexity of modern software projects increased, these limitations became more and more evident, together with the tendency of resulting products to be delivered late and to be poorly usable (Patton, 2002). The principles of the Agile Manifesto were then elaborated to inspire how to deliver a high quality product that satisfies the customer while coping with the inevitable changes (in time, resources, requirements, and knowledge) that occur over the course of a

software development project (Williams and Cockburn, 2003). These features have facilitated the widespread adoption of Agile approaches in companies (Cajander et al., 2013); benefits of the adoption were reported in customer collaboration, defect handling, learning through pair programming, thinking ahead for management, estimation, and focus on current work (Dybå and Dingsøyr, 2008).

The rest of this chapter is organised as follows. First, I will position this thesis in the strand of qualitative research that has been of interest both to the CSCW and to the software engineering communities. Then, I will present an overview of the field of the integration of UCD and Agile, starting from a summary of the most recent literature reviews on the topic and highlighting the main issues found there in the integration itself. These issues will be then categorised and discussed based on three themes, or communication breakdowns: user vs customer, role of documentation, synchronisation of iterations. I will then discuss extant literature that suggests how a solution to these issues can be found in the establishment of a suitable organisational environment. Finally, I will focus on the specific case of the integration of UCD and Agile in small companies, discussing literature that advocates training developers on UX techniques as a way to facilitate the integration itself.

2.2 A qualitative research approach to software engineering

The interpretation of the integration of UCD and Agile in the daily practice of organisations can benefit from the lens provided by the highly interdisciplinary field of Computer Supported Cooperative Work (CSCW). CSCW should be conceived of as "an endeavour to understand the nature and requirements of cooperative work with the objective of designing computer-based technologies for cooperative work arrangements" (Schmidt and Bannon, 1992, p. 3). The study of the collaborative work practices characterising software development is at the core of CSCW research because of its complex and interlinked nature (Bjørn et al., 2014). In particular, borrowing the CSCW terminology, the integration of UCD and Agile can be defined as situated in the specific organisational setting where it occurs (Ferreira et al., 2011). This reflects more general remarks on the need to account for the social context before choosing development methods and processes (Cohn et al., 2009; Dittrich et al., 2009); in addition, it reiterates the importance of thoroughly understanding the practical setting on which any proposal for improvement should be grounded (Ferreira et al., 2012), and of involving practitioners in this effort (Ardito et al., 2014). It should be noted in passing that, even though the CSCW perspective is relevant in this work, this thesis does not focus on the computerisation of working practices.

Because of the reasons stated above, this thesis relies on a qualitative research approach, whose main strength is "in exploring and illuminating the in situ practice of software engineering" (Dittrich et al., 2007). A qualitative approach allows to understand issues that are important from the practitioners' point of view (Abelein et al., 2013) and to contribute to knowledge about how methods are interpreted and adapted in specific settings (Dittrich et al., 2007). Software engineering issues to be investigated have become increasingly complex (Dittrich et al., 2009; Dybå et al., 2011). This has led to a growing interest in leveraging qualitative approaches to understand the cooperative practices of design and development and their rationale (Dittrich et al., 2009; Sharp et al., 2016). Results achieved in this way can be "more informative" (Dybå et al., 2011, p. 426) than those obtained through a purely quantitative approach, and are of interest both to the SE and the CSCW communities, which have been recently joining forces to achieve a better understanding of software engineering and of potential solutions to its problems through qualitative methods (Dittrich et al., 2009). In fact, the understanding of human and social aspects that is typical of CSCW can in turn support the main goal of empirical SE, which is "to improve software practice through evidence-driven research" (Sharp et al., 2016, p. 2).

This need for evidence naturally calls for an appreciation of practitioners' points of view (Sharp et al., 2016), involving them in research. This is particularly favoured by the flexibility in research design allowed by qualitative approaches (Ardito et al., 2014; Dittrich et al., 2007). Such involvement brings several advantages, since practitioners' implicit knowledge can inform improvements of software methods (Ardito et al., 2014; Dittrich et al., 2007, 2009; Rönkkö et al., 2005) increasing their sustainability and improving their chances of adoption (Sharp et al., 2016), and can contribute to bridging the persisting gap between research and industry on these themes (Gregory et al., 2016).

Despite all these envisioned benefits and an increasing number of publications on the topic both in the CSCW and SE communities (Dittrich et al., 2009), there is still limited work adopting a qualitative stance over software engineering (Dittrich et al., 2007). A call to researchers in this sense has been reiterated even in recent issues of SE journals (Dybå et al., 2011). It is even suggested that the limited extent of literature applying qualitative approaches to software practice may lead in turn to a lack of awareness and to unfamiliarity about such approaches, in a vicious circle (Sharp et al., 2016). As noted in (Gregory et al., 2016), this strand of research has matured in recent years. Yet, the need for more empirical studies on human and social factors in the Agile development community persists despite repeated claims on the value of sounder collaboration and communication practices (Dittrich et al., 2007) and on the importance of the focus on human and social factors in the implementation of Agile development methods (Dybå and Dingsøyr, 2008). In this context, a qualitative approach is

especially fit because of its emphasis on communication and cooperation (Dittrich et al., 2009), which is also typical of the Agile approach.

In general, this dissertation follows an understanding of software development as "a social process as much as a technical process" (Sharp et al., 2016, p. 8), emphasising the context as much as the product being developed (Cohn et al., 2009): in this perspective, qualitative approaches are particularly well suited to investigate the social and cooperative nature of this activity (Dittrich et al., 2007). As recently noted in (Sharp et al., 2016), the importance of social and human aspects of software practice is well-established: already in 1984, Boehm stated that "personnel attributes and human relations activities provide by far the largest source of opportunity for improving software productivity" (Boehm, 1984), with potentially significant benefits to the industry also in economic terms (Abelein et al., 2013). In addition, major challenges in the Agile community, such as the understanding of cultural change, are recognised to be complex and highly contextual (Gregory et al., 2016). For what concerns the integration of UCD and Agile specifically, extant literature acknowledges that known issues are not merely related to implementation practices, but are rather largely related to interactions within the team, or between the team and customers or management. For instance, Bornoe and Stage (Bornoe and Stage, 2014) list problematic aspects such as developers neglecting usability perspectives, difficulties in establishing credibility in usability engineering and in proving its impact, or organisational factors. Nevertheless, the social dimension of software development, which refers for instance to collaboration, communication, roles and responsibility (Brhel et al., 2015), is still currently under-researched, despite its long and acknowledged importance (Dybå and Dingsøyr, 2008): for instance, Curtis and colleagues (Curtis et al., 1992) report that mainstream software engineering regards as critical "the facilitation of human understanding and communication" (p. 75), but also note that it has received "less attention from the research community than has machine enaction" (p. 77).

In investigating ways to support coordination within the development team, several authors of the CSCW community have suggested the introduction of tools (e.g. (Boden et al., 2014; Guzzi et al., 2015)), especially in the case of distributed teams (Gross et al., 2005); however, tools may only provide a partial solution. For instance, while proposing the prototype of a tool to facilitate coordination around distributed code change management, Halverson and colleagues (Halverson et al., 2006) introduce their work by acknowledging how software development has long been recognised as a domain in which the most difficult issues are beyond technical and largely unsupported to date. In addition, Guzzi and colleagues propose a tool to support coordination in a development team (Guzzi et al., 2015), while pointing out how teamwork in software engineering is problematic, particularly for what concerns awareness and

communication. I incidentally acknowledge here the highly ambiguous nature of the term "awareness" in the CSCW context (Gross et al., 2005), and will in the following refer specifically to group awareness, defined as "the understanding of who is working with you, what they are doing, and how your own actions interact with theirs" (Gutwin et al., 2004, p. 73) in reference to the work in (Dourish and Bellotti, 1992). Group awareness is in fact deemed necessary for collaborative software development due to the complexity and interdependency of software systems (Gutwin et al., 2004).

In summary, software development shows several "soft" issues related to the cooperative nature of design and development practices (Dittrich et al., 2009). Developers spend a large share of their time communicating with colleagues (Grinter, 2003) and performing collaborative activities (Vessey and Sravanapudi, 1995), and team collaboration depends largely on environmental factors (Sarma, 2005). Cooperative practices such as design and development in fact entail articulation work, defined as "the specifics of putting together tasks […] in the service of workflow" (Strauss, 1988, p. 164). Ferreira and colleagues (Ferreira et al., 2011) remark how articulation work is essential particularly in the case of the integration of UCD and Agile, because both approaches are recognised to support cooperative work, which "occurs when multiple actors are required to do the work and therefore are mutually dependent in their work and must coordinate and integrate their individual activities to get the work done" (Schmidt, 1994, p. 7). For instance, UCD techniques such as personas have been acknowledged to support communication among project stakeholders (Guðjónsdóttir and Lindquist, 2008). On the other hand, coordination, cooperation, and collaboration are fundamental features of Agile development that are emphasised and supported by Agile methods and practices (Ferreira et al., 2011; Strode et al., 2012). This is due to the intrinsic inclination of such methods towards constant collaboration, which in turn facilitates mutual understanding and trust (Hasnain et al., 2013; Kollmann et al., 2009; Pikkarainen et al., 2008). In addition, the effectiveness of the Scrum approach has been found to be also due to its capability to support communication and coordination (Pries-Heje and Pries-Heje, 2011).

2.3 Integration of user-centred design and Agile development

In the last decade, several authors have explicitly acknowledged the similarities of UCD and Agile. For instance, (Da Silva et al., 2011) noted that since Agile assumes a close connection with users, and UCD assumes rapid iterations of the design with users, the two approaches show a natural fit. Similarly, Sohaib and Khan underline that "usability focuses on how the end users will work with the software and agile development focuses on how the software should be developed" (Sohaib and Khan, 2010, p. 32). Following this promising match, there have

been several attempts at integrating UCD and Agile, leveraging the similarities of the two methods while mitigating their differences as reported e.g. in (Chamberlain et al., 2006): the aim is to achieve a more holistic software engineering approach and in turn to support "the execution of software development projects targeting the delivery of useful *and* usable software" (Brhel et al., 2015, p. 177).

For example, an approach called *Agile usability* is proposed in (Wolkerstorfer, 2008), enriching the Extreme Programming methodology (Beck, 2000) with several artefacts drawn from Human-Computer Interaction, 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 (McGinn and Chang, 2013). Others propose to lighten UCD approaches in order to keep the pace of Agile iterations (Memmel et al., 2007): 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 (Sohaib and Khan, 2010). Ungar and White describe a case study in which the application of the design studio approach might effectively bridge UCD methods and Agile ones, such as Scrum (Schwaber and Sutherland, 2011): this approach envisages a rapid, iterative process of ideas generation, discussion, and reconciliation into a unique design concept to be implemented (Ungar and White, 2008).

Besides these examples, the growing interest in the integration of UCD and Agile can be witnessed in the large amount of papers surveyed in the variety of literature reviews on the topic. In the following, I will summarise the findings of the most recent ones.

2.3.1 State of the art in the integration

Sohaib and Khan underline that the way in which the perspectives of usability and Agile are integrated in practice is still not well understood (Sohaib and Khan, 2010). In their review, they focus on identifying the tensions between them, resulting in the following list:

- Customer vs end-user focus: Agile approaches do not explicitly incorporate usability engineering practices, hence they have difficulties in handling user-centred requirements; furthermore, the user is often replaced with the customer;
- Working software vs usable software: even though working software is an important measure of project success, being also cited in the Agile manifesto (Beck, 2001), the two concepts are not equivalent;
- Required design vs upfront design: there is an open discussion on how much design should be performed before development begins;

- Unit testing vs usability testing: Agile methods do not include practices that directly support usability testing, nor allow extensive time to perform it.

Nonetheless, Sohaib and Khan conclude that usability engineering and Agile fit well with each other, and that a number approaches can be attempted to better adapt the former to the needs of the latter.

Da Silva and colleagues performed a systematic literature review of 58 papers to investigate how usability issues are addressed in Agile (Da Silva et al., 2011). They identify a set of practices, artefacts, and needs common to surveyed papers, and highlight the following key aspects of the integration of UCD and Agile:

- Little design up front: while there is consensus that extensive design activities before development are not feasible in an Agile setting, still there is agreement that most interaction design must be done up front;
- Prototyping: prototypes can effectively support communication among developers, UCD specialists, and customers; in addition, they can be used for usability evaluation and as templates for development;
- User testing: there are different opinions on the techniques and the timing that are most appropriate for user testing;
- User stories: there is no consensus on how user stories should be originated, and on how they should be used; however, it is suggested that UCD specialists should be trained in writing them in a technical-aware manner to facilitate communication with developers;
- Usability inspection: when performed by developers, usability inspections change their understanding of the UCD specialists' work, shifting the emphasis from the quality of the code to its impact on users' life;
- One sprint ahead: there is consensus on design needing to be one or more sprints ahead of development, and possibly aligned with business objectives.

Authors conclude that there is a clear need for more empirical studies, because most surveyed papers presented UCD and Agile as a natural fit, yet only provided lessons learned or experience reports as evidence of this. However, the situation has not improved in a later update of the study (Da Silva et al., 2015): authors here performed a systematic mapping of 46 papers on Agile UCD published in the most prominent Agile and HCI conferences between 2003 and 2014. They found that papers on the integration of the two approaches are only occasionally published in major HCI venues; in addition, the majority of papers still consisted of experience reports. Nonetheless, authors suggest that this could be due to the suitability of "real world"

research settings for this topic, given that a basic structure for the integration seems to be now shared among researchers and practitioners.

In 2014, Jurca and Salah performed two literature reviews (Jurca et al., 2014; Salah et al., 2014), 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. Jurca and colleagues performed a systematic mapping study to identify gaps in existing literature, and report the following major trends:

- organisational pitfalls of Agile-UX integration, such as power struggles between designers and developers, or lack of a UX vision for the product, which do not however diminish the overwhelming evidence on Agile and UX being a strong fit for each other;
- organisational recommendations, such as allowing sufficient time to UX designers, sharing the workspace between designers and developers, and emphasising organisational support for UX;
- artefacts and practices, which should be lightweight and easily accessible;
- integration frameworks.

On the other hand, Salah and colleagues focused on the challenges that may affect the integration of UCD and Agile, while listing the practices that have been tried to address them. Such challenges were grouped as follows: lack of allocated time for upfront activities; difficulty of modularisation; optimising the work dynamics between developers and UCD practitioners; performing usability testing; lack of documentation. Results of both reviews agree with (Sohaib and Khan, 2010) in suggesting that much literature on the topic consists of experience reports, lacking rigorous studies or systematic guidelines, and even resulting in contradictory advice, as also remarked in (Brhel et al., 2015; Ferreira et al., 2012). Moreover, that research is dispersed over a large number of venues and is likely to not fully reach its intended target community. Jurca and Salah therefore conclude that there is a need for design and development methodologies that can "draw on the best practices and tools of the two disciplines", as already pointed out in (Sohaib and Khan, 2010, p. 35).

The systematic literature review proposed by Brhel and colleagues is the most recent one considered here (Brhel et al., 2015). It aims at investigating the principles underlying an integrated approach, thereby abstracting from the results of previous literature reviews. Authors reviewed 83 papers and coded them based on the following dimensions: process, practices, people / social, and technology. Interestingly, authors were not able to extract principles for the people / social dimension, because recommendations proposed by surveyed papers were

contradictory. Authors therefore resorted to highlighting the importance of contextual factors such as organisational support and cultural change in a successful integration of UCD and Agile, but remark the need for further empirical research, in line with the recommendations in (Dybå and Dingsøyr, 2008; Jurca et al., 2014; Salah et al., 2014). Other authors go further in this call, explicitly advocating more action research on the integration of UCD and Agile (Brhel et al., 2015; Da Silva et al., 2011, 2015); this dissertation contributes to filling this gap.

In conclusion, all of the literature reviews analysed above agree in consistently identifying a good fit for UCD and Agile in surveyed papers and a need for more empirical studies on the topic. The same papers however also highlight critical challenges in the integration of the two approaches, with frequent overlapping across authors. These challenges can be summarised as follows:

- whether the focus should lean towards the customer, i.e. towards working software and unit testing, or towards the user, i.e. towards usable software and usability testing;
- what artefacts and practices can support communication among developers, designers, and customers;
- how much design should be performed upfront, how much time should be allocated to it, and how should design and development coordinate;
- how the organisation should change to accommodate and effectively support the integration of UCD and Agile.

2.3.2 Communication breakdowns

This set of challenges demonstrates that the integration of UCD and Agile is not trivial to achieve and implement, despite the strong complementarities of the two approaches (Brhel et al., 2015; Da Silva et al., 2011, 2015). In this section, I will examine these challenges more in detail based on examples drawn from literature. In order to facilitate such discussion, however, I would like to first introduce the concept of communication breakdown, which is a "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" (Bjørn and Ngwenyama, 2009, p. 232). Although originally discussed with respect to global software development, a field whose goals are summarised in (Bjørn et al., 2014), this concept refers to issues that are due to an "underdeveloped shared context of meaning" (Bjørn and Ngwenyama, 2009, p. 231). Borrowing the terminology in (Clark and Brennan, 1991), communication breakdowns can also be understood as the incomplete establishment of a common ground

between designers and developers of the same company, or interpreted as the pieces of incoherence in two approaches (UCD and Agile) which would otherwise fit well together.

In the same paper, Bjørn and Ngwenyama remark that the foundation for shared meaning, and therefore for the resolution of communication breakdowns, is the organisational context, in which they identify three analytical categories, each corresponding to a different level of shared meaning:

- the **work practice** level concerns the practicalities of work, such as profession-specific collaborative practices and languages. Communication breakdowns manifest at this level, questioning the efficacy of practices, even though their root cause may lie in other levels;
- the **organisational structure** level refers to explicit and articulated structures, such as policies and methodologies;
- the **lifeworld** level concerns collective experience, tacit knowledge and culture, and in general a frame of reference made of beliefs and values that enables individuals to interpret situations.

In this framing, the resolution of communication breakdowns may require different types of re-assessment at each of the three levels. In particular, given the relevance of the organisational context for establishing a shared meaning as stated by Bjørn and Ngwenyama, and for effectively supporting the integration of UCD and Agile as noted in literature reviews summarised above, resolving communication breakdowns requires a re-assessment of policies and procedures at the organisational context level, and of mental models at the lifeworld level.

In terms of the four challenges listed at the end of section 2.3.1, this relates to the last challenge, which concerns how the organisation should change to effectively accommodate the integration of UCD and Agile. For what concerns the first three challenges instead, namely user vs. customer; role of documentation; synchronisation of iterations, they represent the critical themes that emerged from an analysis of literature reviews on the topic, and can be considered as communication breakdowns. The concept of communication breakdown can in fact serve as a suitable structuring device to classify existing literature that identifies difficulties in the integration of UCD and Agile, as I will show in the next subsections. In addition, the four challenges can be regarded as sensitising concepts (Blumer, 1954), i.e. as providing a general guidance in approaching empirical instances: more details on this definition can be found in Chapter 3.

2.3.2.1 User and customer

The first communication breakdown relates to the differentiation between the user and the customer, and to the extent of their involvement in the design and development process. As mentioned in section 2.1.1, the term "involvement" is here intended as one of the dimensions of user-centredness (Iivari and Iivari, 2011), and qualifies 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) (Damodaran, 1996). User involvement in the informative and 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 by allowing people to shape the tools which will affect their work or personal life (Bjerknes and Bratteteig, 1995).

In UCD, user-centredness is typically described in this continuum from involvement to participation (Iivari and Iivari, 2011): 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. On the other hand, Agile approaches focus on 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 (Ambler, 2015a), 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. (Kane, 2003; Martin et al., 2004; Schwartz, 2014)). However, (Chamberlain et al., 2006; Sharp et al., 2008) report that it is infrequent for a real end-user to act as a customer, subsequently questioning the extent to which the customer can actually represent the real user and his/her needs (Beyer et al., 2004; Sohaib and Khan, 2010). Others recommend that the customer's engagement should also be supported by a number of other roles within the team, such as a proxy user (Chamberlain et al., 2006); alternatively, as customers are part of the release planning and iterative development process (Beyer et al., 2004), their role could be filled by one or more members of the product team (Sy, 2007). The duties of the customer include acting as the voice of the end user, evaluating performance, and helping to prioritise and plan cycles and releases (Schwaber and Sutherland, 2011). Martin and colleagues (Martin et al., 2004) note that this responsibility can turn out to be overwhelming 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 (Beyer et al., 2004; Schwartz, 2013a).

Indeed, the very same capability of users to articulate their own working practices or to design a system can be questioned (Beyer et al., 2004); 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 (Beyer et al., 2004; Gregory, 2003). 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.

2.3.2.2 Role of 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 approaches 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 between designers and developers, such as scenarios or personas, Agile tends to emphasise face-to-face informal communication. Therefore, a second communication breakdown concerns the role of documentation in the two approaches.

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 (Rogers et al., 2011). 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 (McInerney and Maurer, 2005) authors remark that in Agile development the use of documentation is diminished, to the point that one of the principles of the Agile manifesto states that "the most efficient and effective method of conveying information to and within a development team is face-to-face conversation" (Beck, 2001). In (Brown et al., 2011), 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. In fact, 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 (Selic, 2009): the argument is hence about what is to be documented (Selic, 2009), and how to document it and for what purposes, such as supporting organisational memory and communicating with stakeholders (Ambler, 2015b). Nonetheless,

often usability goals are documented in a very general way, relying on a common oral understanding instead (Cajander et al., 2013): this may however make a quantitative evaluation of such goals problematic and make the fulfilment of the big picture of UX more difficult (Lárusdóttir et al., 2012).

2.3.2.3 Synchronisation of iterations

The third communication breakdown is about how to synchronise the pace of UCD and Agile (Jurca et al., 2014; Salah et al., 2014), and in particular whether the two approaches should proceed in parallel or not. Several proposals envision designers and developers working closely together in a synchronised manner. For example, in (Miller, 2005) a daily interaction between them is defined as essential for a successful outcome of the project; in (Brown et al., 2011), their collaboration is defined as informal, oral and ad-hoc. Schwartz found that the development pace was better maintained in a project with an 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 (Schwartz, 2013a).

Other researchers propose to keep UCD and Agile separate instead, while just synchronising their periods of iteration so that design stays ahead of development (Chamberlain et al., 2006). This is the interpretation of Agile usability given by Nodder and Nielsen (Nodder and Nielsen, 2010), 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 (Sy, 2007) 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 (Miller, 2005): 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 (Fox et al., 2008). UCD encourages the team to understand their users as much as possible before writing code (Chamberlain et al., 2006): 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

(Chamberlain et al., 2006): they capture user stories instead, that is high-level requirements to be addressed at the beginning of each iteration (Ambler, 2015a), therefore reducing upfront work to a minimum. Still, several authors (e.g. (Salah et al., 2014; Sy, 2007)) advocate a solid understanding of the user also in terms of time and effectiveness and suggest the relevance of an "Iteration 0" 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 (Lárusdóttir et al., 2012).

2.3.3 Conclusion

This chapter has positioned this work in the strand of qualitative research that is of interest both to CSCW and SE. In addition, a review of existing literature on the integration of UCD and Agile has highlighted four challenges in the integration itself. Three of these challenges have been defined as communication breakdowns, namely user vs customer, role of documentation, and synchronisation of iterations. The definition of these three communication breakdowns will be refined through the empirical findings of the studies described in Chapters 4 and 5. The last challenge concerns instead how the organisation should change to effectively accommodate the integration of UCD and Agile: literature suggesting how small companies specifically may support the integration by training developers on UX will be reviewed in Chapter 6, while Chapters 7 and 8 will present two studies addressing this topic.

3 Methodology

This chapter describes the methodology followed in this thesis. It starts from motivating why this work is positioned within qualitative research; it then introduces sensitising concepts as a structuring and interpretive device used throughout the dissertation. The two stages of the thesis are then described: the first one is dedicated to understanding practice, the second one to improving it. The research strategies adopted in each stage are also explained and motivated: ethnographically-informed studies in the understanding practice stage, action research (and its specific adaptation called Cooperative Method Development) in the improving practice stage. Finally, semi-structured interviews and thematic analysis are described and discussed as techniques of data collection and analysis respectively, used across all of the four studies presented in this thesis.

3.1 About qualitative research

As explained in section 2.2, this thesis is positioned in the growing area of research on understanding the cooperative practice of design and development and its rationale through qualitative approaches (Dittrich et al., 2009). Qualitative research is characterised by a preference for data such as words and images rather than numbers; for documenting reality from the point of view of people studied; for naturally occurring data (thus favouring observation over experiments and unstructured interviews over structured ones); and for hypothesis-generating rather than hypothesis-validating research (Silverman, 2013). Marshall and Rossman report that a qualitative approach stresses the importance of context, setting, and participants' frames of reference; it is particularly suitable for research delving in depth into processes, and for investigating real organisational practices in contrast to stated policies (Marshall and Rossman, 2014). Such research cannot however take place in a vacuum: human actions are significantly influenced by the setting in which they occur, which entails a complexity of norms, roles and values. Research on practices should therefore be performed in real-life situations, i.e. in the setting where the actions of interest take place, capturing the point of view of participants through face-to-face interaction (Abelein et al., 2013; Marshall and Rossman, 2014).

Qualitative research has sometimes been criticised with respect to quantitative approaches because of concerns on the reliability and the validity of its findings. The first concern refers to ensuring the consistency in the categorisation of the events described, and can be addressed by thoroughly documenting the research procedure followed and its evaluation criteria; the second concern refers to avoiding "anecdotalism", i.e. the selection of particularly fitting instances of

the data to ground interpretations, and can be addressed by engaging in a discussion of evidence for and against the researcher's arguments (Silverman, 2013).

This thesis relies on the understanding of the practical setting in which development processes take place (Ferreira et al., 2012), and aims at putting forward proposals for their improvement: to this end, a qualitative approach seemed more appropriate than a quantitative one because of the characteristics of qualitative research summarised above. This stance makes the results of this thesis relevant both to the software engineering and the CSCW communities, which have been recently joining forces in leveraging qualitative methods "to understand software engineering better and make some contribution towards an assessment of its problems and putative solutions" (Dittrich et al., 2009, p. 394). Indeed, the understanding of human and social aspects that is typical of CSCW and of its emphasis on ethnography can inform an improvement of software methods (Ardito et al., 2014; Dittrich et al., 2007), achieving one of the main goals of software engineering, and enable "to see how to better coordinate a highly complex design and development activity where several stakeholders are involved" (Rönkkö et al., 2005, p. 434).

3.2 Research design

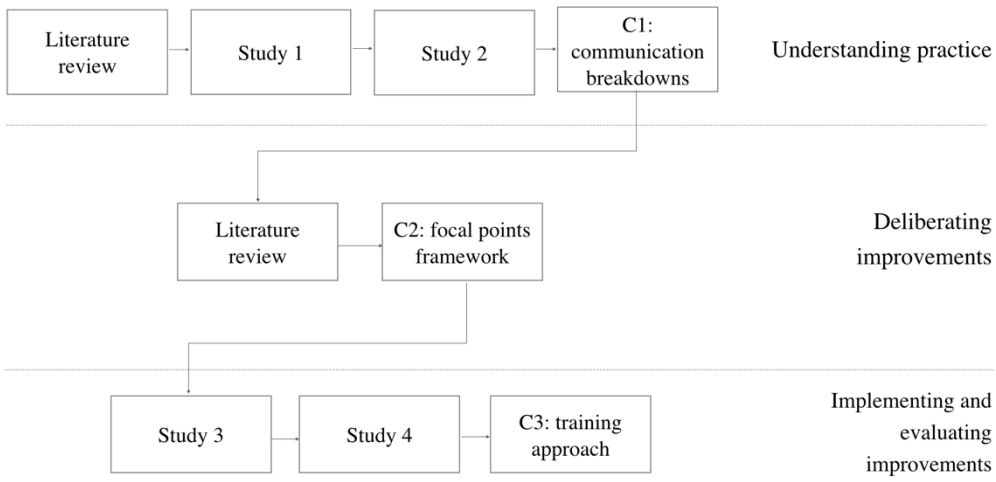


Figure 3 - dissertation outline

As explained in section 2.3.2, the analysis of literature allowed to identify four challenges in the integration of UCD and Agile, which I have used as sensitising concepts for the analysis of the empirical parts of this thesis (Blumer, 1954). According to Blumer, a sensitising concept "gives the user a sense of reference and guidance in approaching empirical instances" (p. 7),

whereas a definitive concept provides a precise definition that serves to clearly identify any instance of the class covered by the concept. This definition is relevant to this thesis because a sensitising concept provides a general orientation to the kinds of data that are of interest to the researcher (Kuczynski and Daly, 2003), or in other words suggests the researcher "directions along which to look" (Blumer, 1954, p.7). It can therefore accommodate the variety of instances of the same phenomenon through progressive refinements of what is covered by the concept itself in light of empirical findings.

In this thesis, sensitising concepts were identified from literature: as commented in (Kuczynski and Daly, 2003), "existing theoretical ideas play a generative role and serve to orient the researcher to the phenomenon that is to be explored" (p. 385). These sensitising concepts informed data collection and analysis as described in section 3.5, serving as interpretive devices to lay the foundation for studying the data (Bowen, 2006); in turn, this allowed to gradually refine the concepts themselves as it will be described in sections 4.3, 5.4, and 6.2. Therefore, the thesis has followed a sensitising approach. Its enactment was supported by the choice of a flexible research design, in which key parameters may be changed during the course of the study to be adjusted to the findings that successively emerge (Runeson et al., 2012). A flexible research design is in contrast to the traditionally fixed design of experiments, which would have been inappropriate in this sort of research because of the complexity and unpredictability of investigated real-life settings as already noted in section 3.1.

Figure 3 shows the three stages that compose the research described in this thesis, which mirror the influence of the CSCW and software engineering communities: the first stage is about building an understanding of the setting of practice, as relevant to the CSCW community; the second and third stages are about deliberating and implementing improvements to development methods, as relevant to the software engineering community. All stages are grounded in practice, as recommended e.g. in (Eriksson et al., 2009) or as follows: "in order to optimize the impact of usability and UX on software development, it is fundamental to analyse current development practices, involving practitioners and possibly working from inside the companies" (Ardito et al., 2014, p.543). Four studies were carried out overall: two belonged to the understanding practice stage, and two to the implementing improvements stage. Each study was performed in a different company: even though each organisation is unique, qualitative findings and analytical results obtained from each study may be applicable to other settings (Eriksson et al., 2009; Sharp et al., 2016). Sections 3.3 and 3.4 will detail the methodologies adopted in the understanding practice stage and in the implementing improvements stage respectively.

3.3 Understanding practice: ethnographically-informed studies

The understanding practice phase began with a literature review that led to the identification of challenges in the integration of UCD and Agile, which were then used as sensitising concepts to guide subsequent activities and consolidate the definition of communication breakdowns, as mentioned in section 3.2. The rest of this stage consisted of two ethnographically-informed studies performed in small companies where such integration was already in place. I adopted the interpretation of ethnographically-informed study provided in (Robinson et al., 2007), whose core aspects are the following:

- The nature of practice is not known a priori, thus its actuality needs to be ascertained empirically through studies driven by research questions rather than by hypotheses;
- Data are collected as they are naturally occurring, not attempting to change or influence practice during studies; resulting accounts draw out perspectives that the developers recognise as valid.

Ethnographically-informed studies adapt traditional ethnography to settings of practice in which an extensive immersion is not possible because of time or confidentiality constraints. This results in shorter studies that fit more easily with a product development cycle. The viewpoint of this research strategy, however, remains the same as that of ethnography, featuring the following characteristics (Atkinson and Hammersley, 1994):

- Emphasis on exploring the nature of a particular social phenomenon, rather than setting out to test hypotheses about it;
- Preference for "unstructured" data, i.e. data that were not coded at the moment of data collection based on a closed set of analytic categories;
- Detailed investigation of few cases;
- Data analysis that involves an explicit interpretation of the meanings of human actions, mainly resulting in verbal descriptions rather than quantification.

Authors in (Robinson et al., 2007) also note that ethnographically-informed studies are not always fit: in fact, they are not applicable to research questions based on testing hypotheses derived from theory. This is due to the orientation of this research strategy to treating all data as "strange" and to interpreting them in relation to the setting in which they were collected; more in general, this limitation can also be referred to the intrinsic appropriateness of qualitative studies to investigate the "why" and the "how" of phenomena rather than the "what".

In agreeing with (Robinson et al., 2007) on the relevance of a rigorous approach to data collection and analysis to avoid bias, I remark that all findings presented in the two studies are

grounded in the data and were corroborated by checking interpretations with participants: this promoted debates within teams and allowed to consolidate and enhance my understanding of practice. I also underline that since ethnographically-informed studies are research strategies, the methods chosen to concretely instantiate them may vary depending on the contingent setting to be investigated. The specific methods adopted in each of the two studies will be described in sections 4.1.2 and 5.1.1 respectively.

3.4 Improving practice: action research

The second stage of this thesis built on the understanding of practice elaborated in the first stage to deliberate improvements of said practice: this process will be detailed in chapter 6. The third stage of this thesis aimed at implementing deliberated improvements: this was done through two iterations of Cooperative Method Development (CMD) (Dittrich et al., 2008), an adaptation of action research specific to software development settings. I will first introduce here the more general action research approach, referring to the interpretations in (Baskerville and Wood-Harper, 1996; McKay and Marshall, 2001); section 3.4.2 will detail how CMD specialises action research.

3.4.1 Action research

Action research is committed to producing new knowledge by seeking solutions or improvements to real-life practical problems: researchers and practitioners working in the problem context are both engaged in the process and collaborate closely, contributing respectively an intellectual framework and knowledge of process, and knowledge of context (Baskerville and Wood-Harper, 1996). Referring to (Susman, 1983), the same authors describe the cyclical action research process, illustrated in Figure 4, as composed of five phases:

1) Diagnosing: identifying the primary problems that are the underlying causes of the organisation's desire for change and developing a theoretical framework about the nature of the organisation and its problem domain;
2) Action planning: guided by the theoretical framework, researchers and practitioners collaborate in specifying organisational actions that should relieve these primary problems and in establishing the approach to change and the desired target;
3) Action taking: researchers and practitioners collaborate in implementing the planned action, actively intervening into the organisation and causing certain changes to be made;
4) Evaluating: determining whether the theoretical effects of the action were realised, and whether these effects relieved the problems;

5) Specifying learning: disseminating gained knowledge as an ongoing process that includes the following activities: restructuring organisational norms to reflect the new knowledge, leveraging such knowledge for diagnosing in preparation for further iterations, and communicating it to the scientific community.

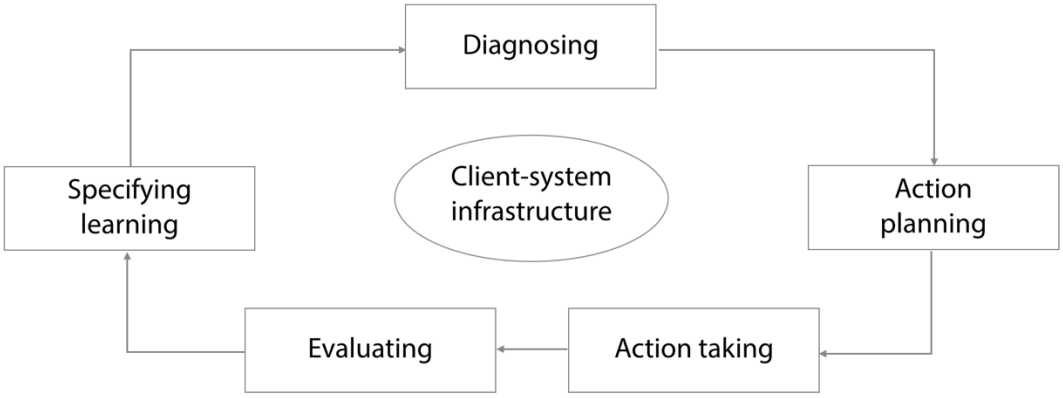


Figure 4 - action research process, adapted from (Susman, 1983)

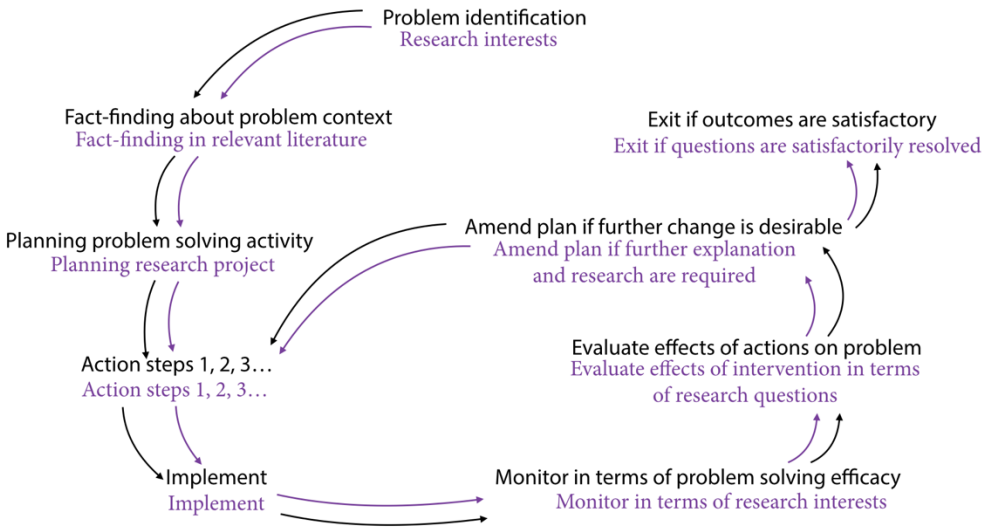


Figure 5 - action research as a dual cycle process, adapted from (McKay and Marshall, 2001)

The action research process relies on the establishment of a suitable infrastructure that legitimates the researcher's intervention and defines the mutual responsibilities of the researcher and of practitioners. Action research therefore has a dual aim of practical problem

solving and theory generation: this is reflected in the conceptualisation proposed in (McKay and Marshall, 2001), which represents the approach as composed of two parallel iterative processes in constant dialogue and are mutually informed by reflection and learning (Figure 5): one process concerns the problem solving interest (pictured in black), while the other concerns the research interest (pictured in purple). This is the conceptualisation to which I will refer in the rest of the thesis: it basically shares the same stages listed in (Baskerville and Wood-Harper, 1996), but highlights the two parallel cycles of problem solving and research in a valuable way.

In terms of the rationale for choosing action research as the most suitable approach for the third stage of the thesis, I started from the comparison of different research strategies that are typical of software engineering research in (Runeson et al., 2012): experiment, survey, case study, and action research. Action research was preferrable because of the flexible design it allows (see section 3.2), the qualitative nature of the data it focuses on (as opposed to the quantitative stance of experiments and surveys), and its primarily improving objective (as opposed to the exploratory character of case studies). More forcefully, Baskerville and Wood-Harper argue that "action research is one of the few valid research approaches that researchers can legitimately employ to study the effects of specific alterations in systems development methodologies" (Baskerville and Wood-Harper, 1996, p. 240). Similarly to Runeson and colleagues, they add that case studies would be unsuitable for these purposes, because introducing methodical changes intrinsically entails intervention: they underline, however, that the researcher should ensure rigour of the intervention itself by suggesting the changes to be done and providing training on them. Furthermore, Baskerville and Wood-Harper claim that "action research, as a research method in the study of human methods, is the most scientifically legitimate approach available" (Baskerville and Wood-Harper, 1996, p. 240) in terms of its relevance to the real world.

Nonetheless, action research is not free from criticisms. First of all, an historical concern claims that action research is actually consulting disguised as research. As remarked in (McKay and Marshall, 2001), action research differs from consultancy because of the research interest cycle: the action researcher has the explicit goal of generating and validating knowledge from actions, and remains constantly cognisant of the chosen theoretical perspective and of the research questions to be answered. Another problem relates to the perceived (yet often factual) lack of rigour: Baskerville and Wood-Harper respond to this with the need to ensure the fitting of research methods to the problem at hand in order to produce valid scientific explanations (Baskerville and Wood-Harper, 1996). In addition, as argued in (Dittrich et al., 2008), the perspective of an external researcher not paid by the company where the intervention takes place can allow for more freedom in time allocation and in the deliberation of actions together

with involved practitioners; this independence also encourages rigorous, unbiased reflection and learning. A final issue relates to the generalisability of the very contingent findings of action research: this can be mitigated by further testing results through a triangulation with other methods (Baskerville and Wood-Harper, 1996).

3.4.2 Cooperative Method Development

As mentioned at the beginning of section 3.4, the third stage of this thesis relied on two iterations of Cooperative Method Development (CMD) with the goal of improving practice. CMD is an adaptation of action research that moves from an ethnographically-inspired understanding of the "existing practice of software development in concrete industrial settings" and aims at improving such practice by cooperating with practitioners (Dittrich et al., 2008, p. 233). CMD is particularly fit with respect to the positioning of this thesis because it "combines qualitative social science fieldwork with problem-oriented method, technique and process improvement" (Dittrich et al., 2008, p. 231). Therefore, it can answer the research questions characteristic of both CSCW and software engineering, described in section 2.2, at the same time, leveraging the understanding of human and social aspects typical of CSCW to support the improvement of software development practice as aimed by SE.

CMD starts from the existing practice of software development in concrete industrial settings: it consists of a cyclic research model based on the action research one (Figure 5) and of a complementary set of guidelines, offering at the same time a framework for supporting research and the flexibility to adapt such framework to the circumstances specific to the project. In summary, the basic cycle underlying CMD is composed of three phases (Dittrich et al., 2008):

- Understanding practice and identifying problematic aspects from the involved practitioners' point of view;
- Deliberating improvements in cooperation between researchers and practitioners;
- Implementing, observing, and evaluating improvements with involved practitioners.

Throughout the cycle, ethnographically-inspired empirical research is emphasised, as well as collaboration with practitioners and the adoption of their perspective: this allows to uncover the rationale of development practices and their relation to contextual factors, particularly focusing on the social and cooperative aspects of software engineering.

3.5 Data collection and analysis

In this section I will describe two techniques that were used across all of the studies that will be presented later in this thesis: semi-structured interviews as a data collection technique, and thematic analysis as a data analysis technique.

3.5.1 Semi-structured interviews

As described in (Runeson et al., 2012), in a semi-structured interview questions are planned as part of an interview guide, but they are not necessarily asked in the same order as they are listed. The development of the conversation in the interview can in fact influence the order in which the different questions are presented to the interviewee, and the researcher can use the list of questions to be certain that all questions are raised with the interviewee at some point. This flexibility allows for improvisation and probing of any issues raised in the conversation.

I used this data collection technique consistently throughout the studies. I preferred it to unstructured interviews, which are more suitable for entirely exploratory studies, because I had already identified sensitising concepts that provided me with some indication of the themes to be covered (see sections 2.3.2 and 3.2); I also preferred semi-structured interviews to fully structured interviews, because I wanted to maintain a flexible research design (as stated in section 3.2) and therefore wanted to retain the freedom of asking unplanned questions in case interesting topics arose during the interview. It is in fact expected that the interviewed participants' viewpoints are more likely to be expressed in a relatively openly designed interview situation than in a standardised interview or a questionnaire (Flick, 2014). Flick also notes that inputs which are characteristic for standardised interview or a questionnaire (and which restrict when, how, and in which sequence topics should be addressed) obscure rather than illuminate the participant's viewpoint. The use of an interview guide is therefore a good compromise between rigour and flexibility: in particular, the consistent use of an interview guide increases the comparability and structuration of the data.

Semi-structured interviews however also require to strike a permanent balance between the course of the interview and the interview guide. It is in fact necessary to constantly mediate between the input of the interview guide and the aims of the research question on the one hand, and the interviewee's style of presentation on the other (Flick, 2014). Consequently, the interviewer must be capable to decide on the spot whether to support digressions and when to cut them, in addition to owning usual interviewing skills such as being a good listener, being able to ask insightful questions and to follow up topics, and being sensitive and non-judgemental.

3.5.2 Thematic analysis

According to (Braun and Clarke, 2006), thematic analysis is a method for identifying, analysing and reporting patterns (i.e. themes) within qualitative data: it can minimally organise and describe a data set in rich detail, and if needed interpret various aspects of the research topic. In using thematic analysis as a data analysis technique across all the studies, I have referred to a "theoretical" interpretation of it, which provides less a rich description of the data overall and more a detailed analysis of some aspects of the data, as opposed to an "inductive" thematic analysis, which codes the data without trying to fit it into a pre-existing coding frame (Braun and Clarke, 2006). Theoretical thematic analysis requires an engagement with literature prior to performing the analysis, and is based on the interchange between the empirical material and prior theoretical knowledge which guides the researcher's attention (Schmidt, 2004). In this thesis, I have followed this approach relying on the sensitising concepts identified from literature.

Braun and Clarke summarise the benefits of thematic analysis, including the following aspects (Braun and Clarke, 2006):

- Flexibility in terms of theoretical underpinning, in how themes can be determined, and in the possibility of generating unanticipated insights;
- Suitability for participatory settings, or in general when participants collaborate with the researcher (which in my case was true in ethnographically-informed studies and even more in action research);
- Capability of summarising key features of a large body of data, highlighting similarities and differences and possibly offering a thick description of the data set.

The analytical process moves from description to interpretation, searching across the whole data set to find repeated patterns of meaning and thus relying on a complete overview of data rather than on single cases or instances. The procedure consists of the following steps (Braun and Clarke, 2006; Schmidt, 2004):

1) Familiarising with the data: data are transcribed if needed (in this thesis, semi-structured interviews were transcribed from recordings) and read several times by the researcher;
2) Generating initial codes: codes identify a feature of the data that appears interesting to the analyst, and refer to the most basic element of raw data that can be assessed meaningfully regarding the phenomenon (Braun and Clarke, 2006). Since this thesis adopted a theoretical approach to thematic analysis, initial codes derived from the

sensitising concepts identified previously. Coding means relating particular passages in the text to one or more codes (Schmidt, 2004);

3) Searching for themes: codes are sorted into potential themes, which capture something important about the data in relation to the research question, and represent a pattern within the data set. Subsequently, all the relevant coded data extracts are collated within the identified themes;
4) Reviewing themes: the researcher goes through the data again to ensure that data within themes are coherent, and that themes can be clearly distinguished; if needed, themes may be revised and adapted;
5) Defining and naming themes: in accordance with the theoretical framework, data extracts for each theme are organised in a coherent account, with an accompanying narrative;
6) Producing the report: this write-up activity should go beyond the description of the data and rather make an argument in relation to research questions, embedding data extracts into a compelling narrative meant to testify the validity of the analysis.

It should be noted that the flexibility of thematic analysis may turn into a disadvantage if the analysis itself is not conducted or organised rigorously, for instance by omitting a theoretical framework of reference in the case of a theoretical stance, or if the researcher is confused about the aspects of the data on which to focus in the case of an inductive stance. In addition, while being able to provide useful insights across the whole data set, it is less suitable for deepening the analysis of an individual account (Braun and Clarke, 2006).

This chapter has discussed the methodologies followed in this thesis. Starting from positioning the work within qualitative research, it described the sensitising approach used throughout the dissertation. The three stages of the thesis, dedicated to understanding practice, deliberating improvements, and implementing and evaluating improvements respectively, were then described, together with the research strategies adopted in each stage and the techniques used across all of the four studies presented in this thesis. In the following chapters I will start describing the three stages more in detail.

4 Study 1: the Smart Campus project

This chapter presents the first study of this thesis. As shown in Figure 6, it is part of the understanding of practice stage, and it is an ethnographically-informed study. The sensitising concepts identified from literature were used to guide the retrospective analysis of data about the Smart Campus project, resulting in a refinement of the sensitising concepts themselves and therefore in a first set of communication breakdowns in the integration of UCD and Agile.

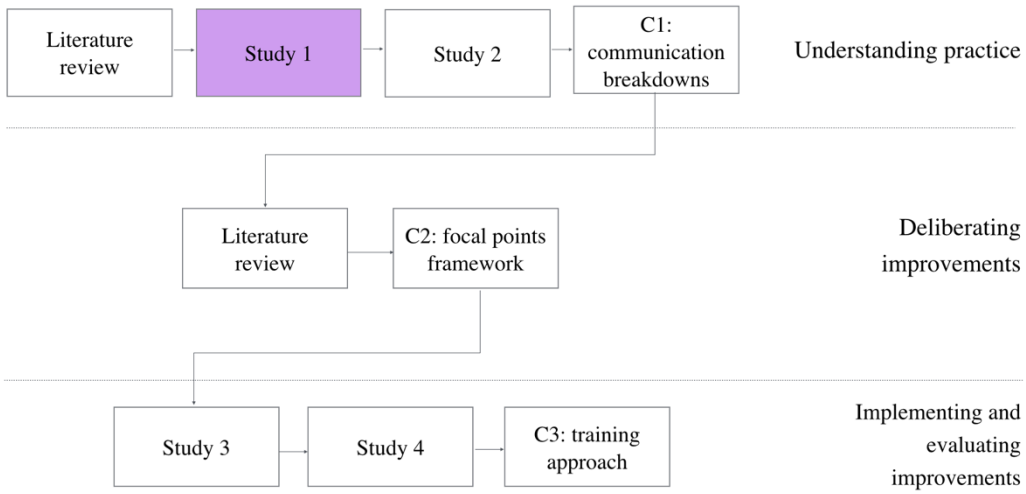


Figure 6 - dissertation progress: Study 1

4.1 The context

The Smart Campus project started in 2012 in the context of establishing a living lab in the Province of Trento and lasted three years. The project aimed at creating an ecosystem that could foster students' active participation in the design and development of services for their own campus (De Angeli et al., 2014a). A service infrastructure was the main technological outcome of the project (De Angeli et al., 2014b, 2014a; Menéndez Blanco et al., 2014); on top of that, a set of eight mobile applications (Figure 7) 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) (Bordin et al., 2014).

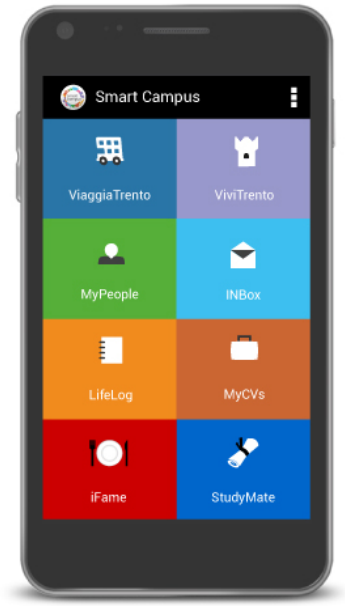


Figure 7 - The Smart Campus mobile application set

The project team consisted of approximately 25 members, including interaction designers and software engineers; furthermore, a number of students of the local university were involved in various ways. In the remainder of this chapter, I will refer to "students" to generically indicate the student community that engaged with the Smart Campus project somehow, e.g. by providing information during the initial requirement elicitation activities (see section 4.1.1). This group includes "users", who downloaded the Smart Campus apps, tested them in their daily life, and possibly provided feedback about them: the number of users varied throughout the project, reaching up to 500 people in some periods. A small number of users entered the project lab as "interns": groups of 4-5 interns at a time were trained as developers, mainly working on the development of the project apps. Each group of interns stayed in the lab for approximately two months; in total, 20 interns were involved in the project.

The communication network (Cataldo and Herbsleb, 2008) representing the actors involved in the project and the artefacts used to support their interactions is illustrated in Figure 8. Approaches for community engagement changed as the socio-technical infrastructure of the project evolved, as discussed in (De Angeli et al., 2014a): some of the users were directly included in the living lab as interns, in a participatory development effort, while others played a consultative role testing the applications as they were developed and commenting on them.

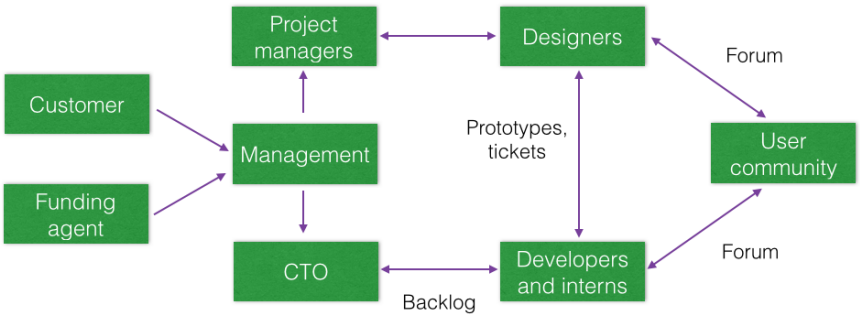


Figure 8 - Communication network in Smart Campus

4.1.1 Project methodology

Figure 9 - Examples of low-fidelity prototypes

During the first year of the project, design and development proceeded on two parallel tracks: design focused on conceptual work and community engagement following a UCD approach, while development focused on the service infrastructure. Several design activities such as focus groups, diaries, online ethnography and workshops engaged students, aiming at investigating the life of the student population and understanding the design space of the project. This information was used to build personas, scenarios, and later storyboards and sketches (Figure 9) that informed the vision of Smart Campus as a toolbox containing separated but interrelated services for students' use, focusing strongly on user needs. 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 set of campus students to seed a user community (De Angeli et al., 2014a). In addition, the dialogue between developers, designers, and users was supported through a set of communication channels including a forum, a GitHub page, and a beta testers community on Google Play.

These initial design 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 Smart Campus 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 (Schwaber, 2004). Since the team had little experience with this approach, the CTO read manuals about how to implement it, involving the project 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 rather 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", i.e. members of the development or management staff who were responsible for a specific app and in charge of tracking its progress, became product owners for different apps; interns were incorporated as on-site user representatives (at least in principle, as discussed later).

The transition to Agile definitively 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. In fact, the design team was supposed to be ahead of the development team by at least one cycle (Sy and Miller, 2008), 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 massive amount of qualitative feedback continuously coming from the user community, prioritising extracted requirements and preparing design solutions to be passed on to the developers for the implementation. The project team attempted a variety of coordination strategies, which will be described later on, but felt that none of them was actually fully satisfactory in integrating design and development activities. As problematic situations kept emerging in the project, a deeper research investigation was organised.

4.1.2 Data collection and analysis

The Smart Campus study was analysed in retrospect, i.e. after the project had already ended, using a rich set of data collected during the project by other researchers with different purposes, as I will explain later; for this reason, data analysis techniques had to be chosen to accommodate available data.

The main sources of data were the following: two interview studies; content analysis of the forum used to communicate with the student community; personal observations of mine, as I worked in Smart Campus as an interaction designer prior to beginning the doctoral programme. In claiming the value of the findings derived from personal observations, I refer to the concept of "autobiographical design" (Neustaedter and Sengers, 2012): while its authors define it as "design research drawing on extensive, genuine usage by those creating or building the system", autobiographical data are intended here as drawing on the same kind of usage, but this time by those creating or building practices rather than a system.

The two interview studies were based on semi-structured interviews (see section 3.5.1) and were both designed, performed, and transcribed by another researcher. Their interview guides are available in the appendix (section 10.1). The 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. It engaged 20 people: 7 staff members (developers and HCI researchers) and 13 students with different levels of involvement with the project. The second interview 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 users involved in the participatory development activities. Information about these two studies are summarised in Table 1.

Table 1 - Interview studies in Smart Campus

Study	Date	Focus	N. participants
First interview study	Summer 2013	Perception of user involvement	20 (7 staff members, 13 students)
Second interview study	Spring 2014	Project documentation and related practices	12 (7 developers, 5 users)

In terms of how the data were analysed, I performed a theoretical thematic analysis (as described in section 3.5.2) that was guided by the sensitising concepts I had by then identified from literature (see sections 2.3.2 and 3.2). In analysing the data, I sought to answer RQ1: in which aspects of work practice do difficulties in integrating UCD and Agile appear? (see section 1.2). This was done by looking for empirical instances of the sensitising concepts in the data (Bowen, 2006). In this case, an alternative analysis technique such as grounded theory was discarded because I had been immersed in the project myself, and was not able therefore to detach completely from my own experience, and because sensitising concepts already provided a theoretical basis for the analysis. Nonetheless, care was devoted to maintaining an open mindset that could anyway consider any unanticipated themes identified in the data in order to refine and enrich the sensitising concepts available.

4.2 Results

This section presents the outcomes of the thematic analysis: as stated in section 3.5.2, the write-up activity goes beyond the description of data and makes an argument with respect to research questions, embedding data extracts into a compelling narrative (Braun and Clarke, 2006). Because of this, the following paragraphs will be structured based on the identified sensitising concepts representing communication breakdowns in the integration of UCD and Agile, namely user vs. customer, role of documentation, and synchronisation of iterations. These paragraphs will also illustrate the number of attempts at instantiating work practices in Smart Campus to put in place such integration. Quotes extracted from the interviews will be attributed to interviewees as follows: *Dev* for developers; *Des* for designers; *Int* for interns; *Stud* for students in the community.

4.2.1 User vs. customer

The user and the customer were clearly differentiated in Smart Campus, as shown in Figure 8: the project followed the understanding of (Chamberlain et al., 2006; Sharp et al., 2008), according to which these figures denote different targets of design interventions. The local University acted as the customer of the project: the relationship between its representatives and the Smart Campus management remained however difficult throughout the project, to the point that as soon as the fundings body ceased its financial support, the University disengaged from the project outcomes, showing no interest in further supporting the user community or the apps development. This reinforces considerations in (Bjørn and Ngwenyama, 2009) about the organisational structure being the main responsible for the failure or success of working practices. Concerning users instead, their involvement in the Smart Campus project ranged along the continuum between involvement and participation (Iivari and Iivari, 2011), evolving

throughout the project in an attempt to transform campus students from users to a community of service developers (De Angeli et al., 2014a). 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.

Nonetheless, 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. Throughout the project, in fact, such meaning appeared to differ between designers and developers. In particular, this misalignment in understanding was exacerbated by the interaction with a community rather than with a well-defined user: this became evident at a management meeting in April 2013. The discussion regarded the case of a student who was particularly active in the project 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 was not always smooth, especially over the forum:

Dev2: "There are different kinds of users: on the one hand they are really careful or even shy and on the other hand they are kind of arrogant […] I prefer not to answer and 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, leading to the organisation of the first interview study. All members of the Smart Campus team reported being aware of the project aiming towards participatory development (De Angeli et al., 2014a) 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 (De Angeli et al., 2014a): 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 can not listen to every little thing from the students."

These quotations show that the perception of the community role retained some UCD 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 user 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 of limited engagement was evident also in the project 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. In fact, users' posts seldom focused on proposals for improvement: hence,

developers increasingly consolidated a perception of the community as a group of testers, and did not entirely commit to acknowledging the actual suggestions for functionality enhancement; as a result, such kind of contributions seemed to appear less over time.

4.2.2 Role of 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 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" "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 it 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; these 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, interviews showed how a series of attempts at effectively supporting documentation resulted in contrasting outcomes. The following paragraphs summarise what happened with reference to different 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, the forum 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 perceived it just as a unidirectional informative tool from the users to the lab and did not consider 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"

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." "[The forum] is 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."

4.2.3 Synchronisation of design and development

This section analyses issues that emerged in the synchronisation of design and development based, rather than on interviews, on the work practices tried throughout the project to suitably coordinate the design and the development team, effectively accommodating the feedback coming from the community while not impeding the development pace. Such approaches are described in the following, classified according to their degree of structure.

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 such 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 choice however 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. The semi-formal approach includes different kinds of meetings: notably, none of them directly involved any user representatives with the exception of the interns. *HCI meetings* usually gathered the whole design team and the champions of the apps. *Matchmaking meetings* were usually held once every two 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* corresponded to the daily meetings envisioned by the Scrum approach (Schwaber and

Sutherland, 2011) and replaced matchmaking ones when the team transitioned to Agile: these quick daily meetings aimed in fact 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 the apps right before a major release. 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.

Another problematic issue was related to the interaction with a whole community of users, which caused the amount of user feedback to escalate quickly: this factor seems to have been one of the most disruptive in an effective synchronisation of the periods of UCD and Agile. This confirms findings of the software engineering community (e.g. (Gartner and Schneider, 2012; Lee and Ko, 2012)), where feedback management and prioritisation are often mentioned 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 however 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: "In order to check the status […] of some application that I am developing […], 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 (Chamberlain et al., 2006), 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 Smart Campus 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; 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.

4.3 Discussion

As explained in section 2.3.2, a literature review allowed to identify a set of sensitising concepts to support the interpretation of empirical data concerning the integration of UCD and Agile. The first three of them, namely a variable interpretation of user involvement, a mismatch in the value of documentation, and a misalignment in iteration phases were understood as communication breakdowns, and were used in Smart Campus as an analytical frame that reflected current ideas from the literature (Bowen, 2006). This allowed to empirically confirm

the occurrence of these communication breakdowns in a setting, such as the Smart Campus one, similar to a small-to-medium enterprise (SME), hence deepening my understanding of how communication breakdowns manifest in the work practice of such a context (see the categories of the organisational context described in section 2.3.2). In conclusion, the following sections discuss findings from Smart Campus in light of relevant literature on communication breakdowns. I will instead address the sensitising concept about a suitable organisational context in section 4.4.

4.3.1 User vs. customer

Difficulties in defining the role of the Smart Campus user community have affected several aspects of the project, as described previously. Misunderstandings in user involvement have been echoed also in literature: for instance, authors debate on the extent of user/customer involvement both in the UCD (Damodaran, 1996; Iivari and Iivari, 2011) and in the Agile perspectives (Chamberlain et al., 2006). Given the specific orientation of Smart Campus towards a user community, rather than a single user, possible contributions could have derived from participatory design (Gregory, 2003) in 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 in Smart Campus by seeding a user community that could communicate directly with the developers through the forum: the availability of such a platform seemed 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. In addition, informing the design and development process with participatory elements may have helped in retaining the richness and articulation of the voices of the community throughout.

The peculiar context of participatory development (De Angeli et al., 2014a), i.e. integrating users in the development team by having interns in the lab, benefitted 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 (Beyer et al., 2004; Gregory, 2003), such bond with the community became less and less meaningful as students spent time in the lab and adopted more and more the point of view of developers.

As mentioned above, the Smart Campus community was neither perceived as merely representing users, nor as fully representing a proper customer. A user community composed of technically skilled people may however be interpreted as a specific case in which 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. This may also address some of the

concerns expressed in literature about the capabilities of the Agile customer: as mentioned in section 2.3.2**,** in fact, the figure of the customer in Agile is entitled to steer the direction of the project by redefining and re-prioritising his/her own requirements even while development is on-going (Beck, 2001), but at the same time his/her actual representativeness is questioned (Beyer et al., 2004; Schwartz, 2013a; Sohaib and Khan, 2010) 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 (Beyer et al., 2004) instead.

In addition, as the integration between the UCD and Agile methodologies occurs, a compromise is needed between their respective understandings of the user and the customer, both in terms of work practices (Lárusdóttir et al., 2012) 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.

4.3.2 Role of documentation

Based on the experience of Smart Campus, documentation can be kept to a minimum, as encouraged by the Agile principles (Beck, 2001; McInerney and Maurer, 2005), and intended just for internal use only as long as the team is fully co-located. Yet, in a scenario in which users are active participants engaged in participatory development, documentation becomes instrumental (Rogers et al., 2011), especially for coordinating the evolution of an open piece of software. Furthermore, in (Matthiesen et al., 2014) 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 cooperative work.

4.3.3 Synchronisation of design and development

As illustrated above, the amount of user feedback coming from a community can quickly escalate and managing it can affect the smoothness of the development process. Although information loss in feedback management is unavoidable (Lee and Ko, 2012), 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 of its expressiveness as possible. This may also ensure greater transparency over the development process, coherently showing 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.

Yet, 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, in contrast to what reported in (Chamberlain et al., 2006) and suggested in (Beyer et al., 2004; Nodder and Nielsen, 2010). As also shown by several suggestions in literature (e.g. (Schwartz, 2013b)), the management of user input may 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" (Fox et al., 2008), which can even be regarded as exceeding the scope of Agile methods (Schwartz, 2013b). 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. In Smart Campus, for instance, the backend functionalities of the platform were being developed while the user research was being performed. Once the conceptual design is ready, UCD can address the interface design, while Agile can proceed with the implementation of the business logic. In Smart Campus, however, the handover of the conceptual design from this stage, where UCD and Agile proceeded in parallel, to the subsequent iterations, where the two approaches merged, remained problematic.

4.4 Conclusion

As a final remark, and as highlighted in the fourth sensitising concept identified in section 2.3.2, 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 (Cajander et al., 2013; Lárusdóttir et al., 2012). 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 (Cajander et al., 2013; Lárusdóttir et al., 2012): this point will be picked up in the next chapter.

In summary, the Smart Campus study tested and improved the sensitising concepts representing communication breakdowns by means of empirical data. The study had two main limitations:

- Data were analysed retrospectively: because of this, I had to rely on data collected for different purposes than answering my research questions. This for instance resulted in the impossibility of testing the fourth sensitising concept on how the organisational

context should change to effectively accommodate the integration, because data supporting an interpretation of this concept were missing;

- Smart Campus had a peculiar orientation towards a user community that engaged substantially with the design and development teams, while the customer had limited interactions just with the project management. Therefore, the project did not fully represent the SME setting I was interested in, despite being a good approximation of it.

In order to overcome these limitations, the study that will be described in Chapter 5 was organised: it took place in a proper SME where the integration of UCD and Agile was already in place, with the aim of further refining the set of communication breakdowns and therefore reinforcing the understanding of practice stage.

5 Study 2: H-umus

This chapter presents the second study of this thesis. As in the Smart Campus study presented in Chapter 4, and as shown in Figure 10, it is part of the understanding of practice stage, and it is an ethnographically-informed study. The setting was however a proper SME, in which the integration of UCD and Agile was already in place. The analysis was again guided by the sensitising concepts identified from literature and results were then compared with the Smart Campus ones, allowing to consolidate and further elaborate the set of communication breakdowns.

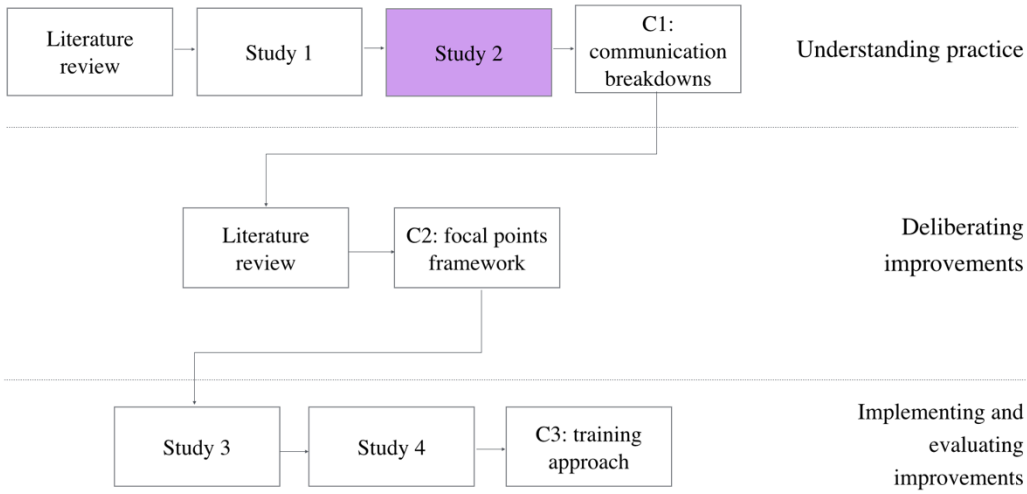


Figure 10 - dissertation progress: study 2

5.1 The context

The study presented here took place in H-umus, presented in their website as a "software and interaction design company". Founded in 2007 in one of the most well known Italian venture incubators, H-umus designs and develops mobile sales tools for the fashion industry and has recently been incorporated into a large Italian software and services business. The personnel include a CEO, a CTO, four project managers (two of whom are also interaction designers), and five developers. At present, H-umus offers two main products to an established customer portfolio: a B2B sales platform and a time and expenses accounting tool. The company also follows some ad-hoc projects for occasional customers, such as the development of a mobile tool for a leading fashion brand that will be called FashionX in the following.

The company adopts a customised version of Scrum. The CTO acts as the Scrum master and developers are divided in two teams whose composition may change dynamically based on contingent necessities. Moreover, each project has a manager, who is usually, yet not always, a designer; the project manager maintains the relationship with the customer and advocates its needs within the project. However, the project manager is not a full-fledged product owner, as he does not e.g. prioritise the backlog. Finally, while retrospectives and sprint planning meetings are run biweekly, daily meetings are not done: the team agreed that co-location in an open space and pair programming, together with the small size of the staff, would allow for sufficient communication anyway. In addition, H-umus follows a loose interaction design approach. Designers are constantly involved in project execution also because they act as project managers; however, user research and testing can often be performed only to a limited extent as explained later, and prototyping is frequently done ad hoc in collaboration with developers.

5.1.1 Data collection and analysis

Table 2 outlines the data collection stage of this study: it consisted of 20 hours of observation of work practices, semi-structured interviews (section 3.5.1), attendance to meetings. In addition to this, artefacts relevant for practice were examined, and semi-structured interviews were transcribed. The interview guide is available in the appendix (section 10.2).

Table 2 – Data collection activities performed in H-umus

Date	Activity	Duration
October 26th, 2015	Attendance of sprint planning meeting; interviews with the CEO, a project manager, a designer and a developer	7 hours
November 20th, 2015	Interviews with both designers and the CTO; observation	6 hours
December 14th, 2015	Attendance of sprint planning meeting; interviews with two developers, a designer, and a project manager	7 hours

Data collection activities were performed by myself. Other methods that were considered for data collection, but were then discarded, were the following:

- Full ethnography: this would have required more time and resources than those available;

- Videotaping: this technique would have looked intrusive, especially given the very coherent nature of the H-umus team and co-location, and would have been violating confidentiality agreements;
- Focus groups: these would have been too much time demanding, especially given the small size of the team; moreover, it could have been hard to get an adequate representation of individual opinions; more in general, it would have been difficult for me to handle group dynamics, moderate the focus group, and document results at the same time while on my own.

I also carried out the data analysis activities, refining results later with my supervisor. As done for Smart Campus, the theoretical thematic analysis (section 3.5.2) was guided by the sensitising concepts identified from literature about communication breakdowns. In this case, my goal was to compare the results obtained from H-umus with those obtained from Smart Campus, refining the answer to RQ1: in which aspects of work practice do difficulties in integrating UCD and Agile appear? Also in this case, grounded theory would not have been an appropriate strategy as I already had a starting theory about the data.

5.1.2 Understanding practice

Collected data allowed to build the communication network (Cataldo and Herbsleb, 2008) in Figure 11, which represents actors working for H-umus or engaging with the company and how, possibly through some artefacts, they are connected. If no label is specified, the arrow indicates that the communication is unstructured or anyway not supported by specific artefacts.

The usual customer for H-umus is a large fashion business, usually represented by its IT department. Customer representatives completely mediate the dialogue with users and communicate with H-umus through a project manager of this company, who is often also an interaction designer; such dialogue is supported by a series of artefacts such as requirements documents, prototypes, and cost or time estimates, which will be described more in detail in later paragraphs. The project manager is then usually the only point of contact between the inside and outside of H-umus: he collaborates with the management (i.e. the CEO) in the early stages of an approach to a new customer, with the CTO in the definition of the technical analysis, and with developers during the implementation. Internal communication is also supported by a range of artefacts. Finally, the owner group refers to the management for products developed on their behalf.

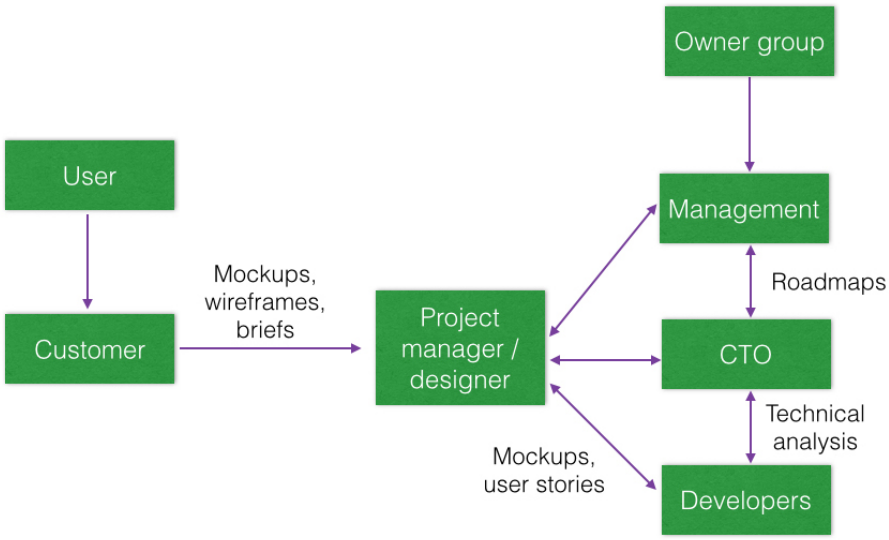


Figure 11 - Communication network in H-umus

In addition, a variety of artefacts are used in H-umus to support communication, both internally and with the customer. The most relevant ones are described in the following.

Mockups and wireframes. In the case of enhancements to already consolidated products, designers prepare high-fidelity mockups relying on the existing interface; in the case of software built from scratch instead, they prepare wireframes, representing interaction flows and layouts. Mockups and wireframes are then iteratively discussed with the customer: this allows to check that requirements have been correctly understood, to ensure that the customer is aware of project status and will not change his mind later, and to skip formal validation steps at the end of each sprint.

Briefs. Prototypes and requirements are integrated in documents called briefs, which crystallise the requirements; they are then iteratively revised with the customer to ensure that both parties share the same understanding of requirements and status of advancement.

Roadmaps. For each project, the relevant project manager keeps a chart showing the evolution of the product at a high level, including milestones to be delivered to the customer. This chart is often linked to other documents reporting, for instance, more extensive descriptions of functionalities or specifications of the customer's target platforms. Roadmaps are used internally, at management level: the CEO, the CTO and project managers refer to them to supervise the status of each project. However, if the customer requires so, roadmaps are also used to provide long-term visibility on the articulation of the project.

Technical analysis. The CTO elaborates this document for each project: it includes finalised interface mockups, a description of the data flow and of the data structure, cost and time estimates, and a finer-grained breakdown of development tasks. The technical analysis serves two purposes: internally, it is a reference for developers to determine what to implement in the next sprints; externally and if needed, it can provide the customer with a detailed understanding of the implementation process.

5.2 Results

This section presents the outcomes of the thematic analysis that concerned the H-umus study, which will the be compared to the findings of Smart Campus in section 5.3. Once more, the thematic analysis was guided by the sensitising concepts identified in literature: the results will therefore be presented in relation to the theme, i.e. the communication breakdown, to which they refer. Quotes in the next subsections have been translated from Italian and will be attributed to interviewees as follows: *Dev* for developers; *Des* for designers; *PM* for project managers who are not designers; *Mgmt* for the CTO and the CEO.

5.2.1 User vs. customer

The distinction between customers and users is very sharp in H-umus and project managers usually communicate only with the customer, who can be represented by different employees at different stages of the same project. Especially when the customer is a large company, its most appropriate representative to liaise with can be difficult to identify and often changes over time:

Dev2: "The most difficult thing in communicating with the customer is understanding who you should be talking to."

In general, the customer representative is the IT department:

Mgmt2: "You would not believe how conservative IT departments can be. Whatever change may affect their working routine, it's a no-no."

There are, however, exceptions to this situation: for example, a few demos were arranged with business and sales representatives of FashionX, i.e. with a sample of final users, in order to collect feedback that could supplement the requirements provided by the IT department of the company. Yet, this only happens occasionally: usually, and as shown in Figure 11, the customer completely mediates user needs, requirements, and feedback. This causes some concern in the H-umus management:

Mgmt2: "Then it is difficult to determine how to handle the feedback we receive and how relevant it actually is with respect to the customer or with respect to the needs users may truly have. […] Sometimes I wonder whom we should really satisfy. Is it the business department or the IT department? We usually speak only to the latter. I believe this causes a large drop in the value we deliver with our products."

H-umus designers acknowledge that it would be desirable to apply a proper user-centred design methodology, involving real users in requirement gathering and interface evaluation. However, this is very hard to achieve in practice, because of two main reasons: first, the time for design is constrained; second, it is difficult to gain access to users. In fact, the customer is not always interested in being actively involved in the design of the commissioned product, or may not be aware of the benefits of user involvement: sometimes H-umus may only be asked to prototype a new graphical interface for an existing software. The customer may even believe that users are not able to provide any sensible contribution:

Dev1: "I do not have any contact with users […] Sometimes they are even described to me as being as dumb as an ox, so it is paramount to design products that are very easy to use, and I guess this is a major challenge for designers."

5.2.2 Role of documentation

The H-umus staff has a small size and is co-located in the same open space: hence, most coordination occurs face to face or at most through instant messaging, both among developers and between developers and designers. This leads to a scarcity of documentation for internal use. However, in order to avoid knowledge gaps in case someone leaves the company, pair programming is adopted when a part of the code needs to be modified: the task is in fact assigned both to the developer who already worked on that code and to a "fresh" developer at the same time. In this way, in the long run everybody will have at least an overview of all the code produced. Working in pairs is also a common practice in the early stages of a new project, where a designer and a developer cooperate in order to shape the design space quickly and based on an understanding of what can be technically feasible.

PM1: "Everybody has an overview, but also a specific responsibility."

Documentation is instead actively and carefully maintained to support the relationship with the customer, not so much as a communication channel, but rather as a device for binding the customer to its requests. Despite the Agile principle of "embracing change" (Beck, 2001), the management highlighted the need of making the customer responsible for his requirements and committed to them. The CTO and the project managers in fact insisted on their strong need to

shield H-umus from sudden, important changes in customer requirements; being the company so small, this could cause a lot of work to be wasted and not paid, causing in turn potentially severe financial issues.

PM1: "H-umus is a small company. If the customer first says he wants a mobile app, and then after six months he comes and says that now he wants a standalone application… We cannot afford that. Unless the customer is paying for the extra time, of course."

Des2: "We do not have much development capacity. It can become a big issue if I draw the mockup and then we have to go back and change fundamental parts of it."

This protection is achieved by using several artefacts that are admittedly not typically Agile: documents such as requirements lists and technical analyses are shared with the customer, iteratively discussed and then signed off.

Mgmt1: "We make the customer sign the requirements document, so nobody can come and say: "This is not what we agreed upon". Whatever extra, we discuss it and it is billed separately."

Des2: "Being able to tell the customer: "Look, this is what we suggested and you approved it" is something that can cover our back when we need to ask for more funding or when we just say that something is not feasible".

The strong perception of documentation as having a purpose mainly in relation to the customer emerges very clearly also in respect to other themes:

Mgmt1: "I'll show you the technical analysis we did for FashionX […] Please write down in your notes that to me this is complete nonsense. The risk estimates and the planning poker and stuff… It is obvious that these numbers are meaningless. Yet the customer wants to have a long-term perspective on the project, so here it is."

5.2.3 Synchronisation of design and development

Given the small size of the company, designers and developers work together, so synchronisation is handled through constant, direct communication. Indeed, there is no separate process for design and for development: for instance, design tasks such as prototyping are listed as regular user stories in the Agile management tool in use:

Des1: "UX aspects are regarded as common functionalities."

Despite a general awareness among the staff of the company transitioning towards a more design-oriented culture, the overall attitude appears to be still strongly technical. For instance, sprint meetings only involve developers:

Mgmt1: "We are born as a data-driven company […] Sprint meetings are too technical; designers would waste time attending them."

Furthermore, a different theme emerges, related to the recognition of designers' expertise in a technically dominant environment. Several times designers referred to their competence in UX as being interpreted as common sense in the company:

Des2: "Why should the CEO's opinion be more relevant than mine, if I designed the interface from the beginning? Sometimes [Des1] and I refer to it as a class conflict with the developers" "Everybody feels entitled to comment on the design, just because each of us is a technology user, while nobody would comment on the code unless competent. So [developers] bring in their own use cases, but we are not developing, say, Instagram, which only has a couple of functionalities: it is totally different. Sometimes the comments are just "I don't like it". I can take it from the customer, if he pays for the extra time needed to rework the design, otherwise I'd expect some sounder feedback."

The rest of the team perceives this issue as well, although in variable ways:

Dev1: "Interfaces are subjective […] usability is subjective too: you need to design stuff that is comfortable for the user, more than functional. [Des1 and Des2] do a great job in my opinion in this respect."

PM1: "The best way to work shouldn't be to tell the designer how to do the things, but just what you need; unfortunately, the customer is often unable to articulate what he wants, and anyway we must give priority to the development to save time."

Dev2: "We all give our opinion, but in the end it is the designer who decides."

5.3 Comparison with Smart Campus

Despite a positive attitude towards UCD, H-umus found objective difficulties in integrating it with Agile in practice. As will be explained in the next paragraphs, some of these difficulties were the same that in Smart Campus contributed to the definition of communication breakdowns. This happened even though the settings of Smart Campus and H-umus differ, as can be seen by comparing the respective communication networks (Figure 8 and Figure 11). The analysis of the H-umus study therefore allowed to consolidate and further refine the set of communication breakdowns as will be detailed in section 5.4.

User vs customer. In Smart Campus, the customer and the user community were two clearly differentiated actors. The local University acted as the customer and its representatives only interacted with the project management; hence, most of the team had direct contact only with

the users through a variety of communication channels such as the forum. However, the perception of user involvement appeared to be variable between designers and developers, denoting an underlying mismatch in the understanding of this concept: while designers struggled to promote a participative role of the user community, developers intended such role as informative or at most consultative instead (Damodaran, 1996). In H-umus, the extent of user involvement remains problematic, although with a different flavour: the customer completely mediates the interaction with the user, so the role of the latter is practically less than informative (Damodaran, 1996). Therefore, it can be argued that the understanding of the extent of user involvement should be shared not only inside the company (among designers, developers, managers), but also outside, by the customer. A related misunderstanding emerged from observations of work practice in H-umus specifically, suggesting that the team should not only share a common language and understanding of user involvement, but also a common understanding about who is responsible for UX tasks: whether designers, because of their expertise and user research, or developers, because of customer or technical constraints. This will help mitigate situations in which a technically predominant environment interprets UX as mere "common sense", which are not conducive to endorsing the added value that UX can provide to a product. To this end, the concept of boundary objects may be helpful: these are mediating artefacts that allow knowledge sharing and promote collaboration since their interpretive flexibility facilitates "an overlap of meaning while preserving sufficient ambiguity" for different groups to read their own meanings (Barrett and Oborn, 2010). The briefs used in H-umus can be considered as boundary objects in this sense, as they gather mockups from designers, technical specs from developers, and business requirements from the customer, and they act as a common reference point for monitoring the evolution of the product.

Role of documentation. In Smart Campus, documentation did not appear to have an intrinsic value as a communication tool for developers; however, it became increasingly relevant to keep the development team aligned when the latter became more distributed due to the introduction of interns working at variable times and often remotely. Yet, how to effectively support the need for a shared knowledge base, particularly referring to design artefacts, remained an open point although the team tried to adopt a variety of articulation platforms. In H-umus instead, the team is co-located: in this case, besides being a tool for tracing the history of the software and the rationale of related design and development choices, documentation can also have an instrumental function in balancing the power relationship with the customer, protecting the company against unsustainable changes in requirements.

Synchronisation of design and development. The Smart Campus project was oriented towards a large and strong user community, whose feedback escalated quickly and was not

mediated (for instance by a customer). This caused severe difficulties in synchronising the iterations of UCD and Agile: designers struggled to elaborate requirements and provide suggestions in a timely manner that could fit the development pace, while developers often took the initiative of fixing interfaces regardless of the overall UX vision. In general, designers resorted to several ad-hoc interventions, elaborated together with the developers requesting them. In H-umus instead, the team is co-located and quite small, so synchronisation can easily occur through face-to-face communication. Furthermore, the existence of signed documents prevents the customer from changing requirements with the same frequency witnessed in Smart Campus with the user community. Yet, several interviewees argued that, in order for an effective communication to occur, it is advisable that the whole team shares a common language.

5.4 Consolidating communication breakdowns

In section 2.3.1 I had listed the following sensitising concepts that reflect the critical issues in the integration of UCD and Agile as found in literature:

- whether the focus should lean towards the customer, i.e. towards working software and unit testing, or towards the user, i.e. towards usable software and usability testing;
- what artefacts and practices can support communication among developers, designers, and customers;
- how much design should be performed upfront, how much time should be allocated to it, and how should design and development coordinate;
- how the organisation should change to accommodate and effectively support the integration of UCD and Agile.

These sensitising concepts were first used to guide the analysis of the Smart Campus study (section 4.2) and then of the H-umus study (section 5.2). During this process, the sensitising concepts were tested, improved, and progressively refined through careful scrutiny of empirical instances occurring in each study, whose features not adequately covered by a concept became a means of revisiting the concept itself (Blumer, 1954). At this point, once the usefulness of the four sensitising concepts was acknowledged, they were ready to answer research question RQ1: "in which aspects of work practice do difficulties in integrating UCD and Agile appear?". Table 3 defines three communication breakdowns in the integration in the integration of UCD and Agile: it constitutes the first contribution (C1) of this dissertation, and closes the understanding practice stage of this research work (Figure 12).

Table 3 – C1: communication breakdowns in the integration of UCD and Agile

Communication breakdown	Description
User vs customer	UCD activities focus on the user, who also drives the overall vision possibly being represented by designers; Agile emphasises constant involvement of the customer, without differentiating it from the user, and expects the customer to direct the overall project vision
Role of documentation	In UCD, it records the overall design vision and its rationale; in Agile, it is minimal and ad-hoc
Synchronisation of design and development	UCD and Agile may proceed in parallel or be integrated into a single process; in addition, UCD emphasises extensive user understanding upfront, while Agile encourages designing just enough at each iteration

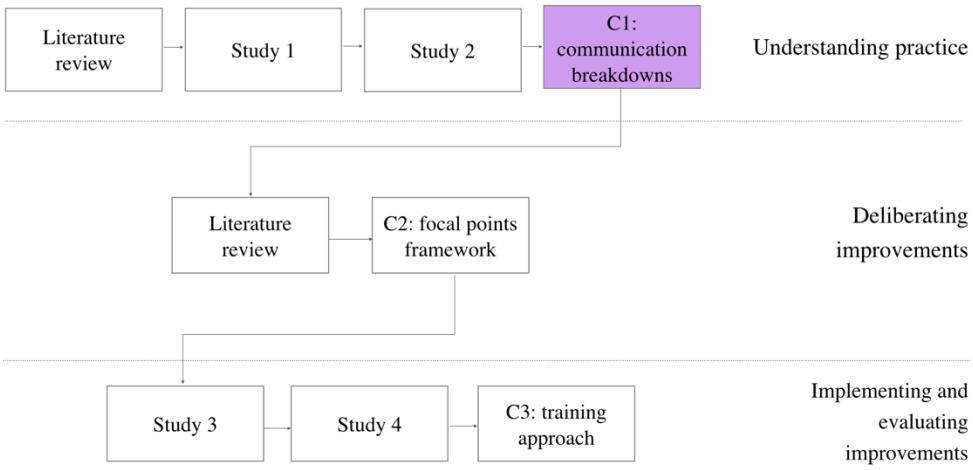


Figure 12 - dissertation progress: contribution C1

6 A diagnostic framework

Referring to section 1.2, the set of communication breakdowns defined in Table 3 constitutes contribution C1, closes the understanding practice stage, and answers the first research question of this thesis (RQ1: "in which aspects of work practice do difficulties in integrating UCD and Agile appear?"). This chapter builds on C1 to answer the second research question (RQ2: "what practical solution can be devised to address the difficulties identified in RQ1?"). This is done by proposing a way to leverage the awareness of communication breakdowns to support an organisation willing to enact the integration of the two approaches.

6.1 State of the art

The set of communication breakdowns was consolidated by using sensitising concepts drawn from literature as an interpretive device for empirical data. In this section, I will discuss additional literature that supported the elaboration of communication breakdowns, which represented potential issues in the integration, into a solution to such issues.

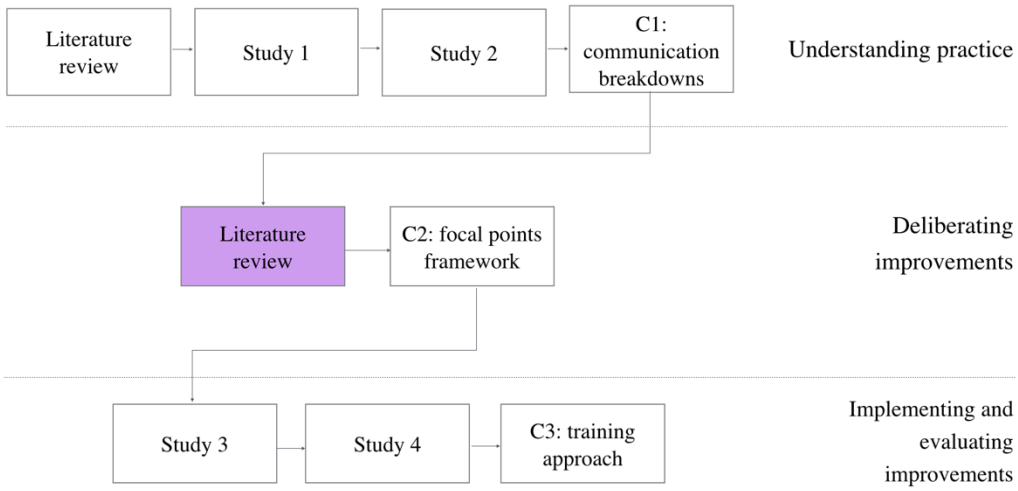


Figure 13 - dissertation progress: additional literature review

I will start from reintroducing the analytical categories of the organisational context listed in (Bjørn and Ngwenyama, 2009) and already presented in section 2.3.2:

- the **work practice** level concerns the practicalities of work, such as profession-specific collaborative practices and languages;

- the **organisational structure** level refers to explicit and articulated structures, such as policies and methodologies;
- the **lifeworld** level concerns collective experience, tacit knowledge and culture, and in general a frame of reference made of beliefs and values that enables individuals to interpret situations.

In their paper, Bjørn and Ngwenyama state that although communication breakdowns manifest at the work practice level, their root cause may lie at the organisational structure level, or at the lifeworld level. Therefore, the resolution of communication breakdowns requires a re-assessment of the different levels of the organisational context of the company where the communication breakdowns appear; in particular, it may require a re-assessment of policies and procedures at the organisational context level, and of mental models at the lifeworld level. In other words, a mitigation of communication breakdowns, and therefore an effective integration of UCD and Agile, relies on the establishment of a suitable organisational environment, as will be illustrated in section 6.1.1: one way to address this, particularly in the resource-limited context of small enterprises, can be training developers on UX and usability, as I will discuss in section 6.1.2.

6.1.1 The need for a receptive organisational environment

As listed at the end of section 2.3.1, the last challenge in the integration of UCD and Agile identified from literature concerns how the organisation should change to effectively support such integration (Brhel et al., 2015; Jurca et al., 2014). Even though communication breakdowns are manifested at the work process level (Bjørn and Ngwenyama, 2009), several authors indeed agree that their solution could be found in a supportive organisational environment (Cajander et al., 2013; Damodaran, 1996; Lárusdóttir et al., 2012): more generally, others also argue that "an organisation cannot evolve beyond its leadership's stage of development" (Laloux, 2014). To leverage the full potential of the integration of UCD and Agile, the management should actively counteract the so-called "developer mindset", i.e. an approach that is overly focused on technical aspects rather than on customer and user satisfaction (Ardito et al., 2014, p. 546): in addition, it should commit to an explicit inclusion of UCD in company goals and financial allocation (Venturi et al., 2006). UX is part of the business strategy and needs to be aligned with the business and product owner teams; moreover, it is not just a practice or discipline within the project, but rather "a way of thinking across the team" (Cho, 2009, p. 421).

Explicitly promoting this understanding entails a cultural change in the company: for instance, practitioners' proposals such as Lean UX (Gothelf, 2013) are acknowledged to rely on a suitable receptive culture (Liikkanen et al., 2014). Such cultural change should in particular be driven by managers and product owners (Brhel et al., 2015): 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 (Cajander et al., 2013; Lárusdóttir et al., 2012). This may also ensure that the overall project vision is not lost because of an excessive focus on details, as risked by Agile approaches (Salah et al., 2014). In order to foster such a receptive organisational culture, one possibility could be the adoption of design thinking, a methodology grounded on a "human-centred design ethos" that holistically pervades all stages of a product lifecycle, from inspiration to ideation to implementation (Brown, 2008). This discipline leverages on "the designer's sensibility and methods to match people's needs with what is technologically feasible" and marketable.

The required cultural change should not be limited to the company, though, but should also engage the customer, in order to win its collaboration in accessing users and allowing for a suitable design to be researched. The lack of sponsorship by a sceptical customer is rather common also in the literature on doing Agile in a non-Agile environment (e.g. (Gregory et al., 2016)): Hoda and colleagues note that it can be a big challenge for development teams (Hoda et al., 2011), especially given the relevance placed on the customer by the Agile philosophy (Coram and Bohner, 2005). Therefore, even after many years from (Nielsen, 1994), the developer mindset needs to be addressed actively, persuading the customer and the technical personnel (at least partially) of the positive cost-benefit trade-off of devoting time to user studies and usability (*The Standish Group's CHAOS Report*, 2014). Practitioners often suggest that the solution requires the capability of demonstrating business value, management support, and nurturing a change of mindset and culture in the customer (Gregory et al., 2016; Kuusinen et al., 2016). However, to this end, a set of actionable measures that can more objectively assess the positive impact of user involvement on the quality of produced software is still lacking (Mao et al., 2005), together with a set of less resource-intensive practices to put such involvement in place (Øvad and Larsen, 2016a).

One of the possible ways to mitigate the communication breakdowns is to train developers on UX and usability, fostering the establishment of a suitable organisational environment: this approach is particularly suitable for the resource-limited context of small enterprises on which this thesis has focused, and will therefore ground the implementing and evaluating

improvements stage of this work. Existing attempts reported in literature at training developers on UX will be discussed in the following section.

6.1.2 Training developers on UX

The need for a set of practices aimed at engaging users while demanding less resources from the company seems to be one of the reasons why "the integration of usability engineering methods into software development life cycles is seldom realized in industrial settings" (Moreno et al., 2013, p. 100), as I will argue in this section. In particular, the following paragraphs will describe three main reasons for this lack of integration in organisations.

First, there is a lack of usability specialists in the industry (Bruun, 2010). This scarcity of available skills in turn leads to a situation where "the major obstacles to the integration of HCD [human-centred design] activities in software development processes are related to the lack of knowledge about what usability is, to the resistance of engineers and/or managers to usability since they do not understand its value, and to the feeling that too many resources (e.g., time and costs) are necessary" (Ardito et al., 2014, p. 543). It is often the case that system developers have an incomplete understanding of the work of the end user and their product is judged by technical quality attributes rather than by its capability to contribute to the end-user's work situation, hence perpetuating the developer mindset (Eriksson et al., 2009).

Second, transferring methods from academia to industry is challenging (Ardito et al., 2014; Brhel et al., 2015). In fact, often proposed usability and UX methods "originate from an academic environment, and thus have not taken the conditions in the industrial setting into account, especially constraints in time and resources" (Øvad and Larsen, 2016a, p. 1080). It is responsibility of academics to translate scientific publications into methods readily applicable to industry (Ardito et al., 2014): "as HCI researchers we should be more concerned about making our methods, tools and knowledge used by practitioners who generate designs and products, and some of these practitioners are developers" (Eriksson et al., 2009, p. 562). In (Ardito et al., 2014), authors also remark that the availability of resource-saving approaches, both in terms of methods proposed and of the effort required by the training itself, is particularly relevant in the case of small enterprises: these in fact do not have the funding to incorporate usability engineering, and are subject to several constraints such as team buy-in, resources, change management, political environments, technology and personnel (Bruun, 2010).

Third, "most usability and UX methods originate from a time when almost all software development followed a waterfall model. [This] entails that the methods are too resource demanding and difficult to apply into today's agile, industrial environments" (Øvad and Larsen, 2016a, p. 1080). Authors in (Sohaib and Khan, 2010) note that this results in a lack of practices

within organisations, which appears to be one of the reasons for the partial achievement of the integration of user needs within the feature-oriented Agile development process. Some companies are however already attempting an adjustment of academic proposals on their own: "industry indeed performs usability and UX work of various complexity and extent. They reveal that methods often diverge from those developed and used in academia, and are adapted towards more informal use" (Øvad and Larsen, 2015, p. 41). In particular, the studies presented in this dissertation were all carried out in companies at stage 3 of Nielsen's usability maturity model, "Skunkworks UX" (Nielsen, 2006). This label means that "the organization realizes that it shouldn't rely on the design team's personal judgment of what will be easy for customers to use" and some grassroots attempts at improving the usability of designs start to appear. This might be the only feasible way for companies at this stage of maturity to perform UX work (Øvad et al., 2015).

Even if a company realises a need to increase the usability and user friendliness of its products, it might be unable to invest in the resources needed to achieve this (Bruun, 2010). In fact, Eriksson and colleagues remark that "the practical reality in many organizations is that usability expertise may not be available, and if that is the case we need to find alternatives. Successful introduction of usability in organizations requires us to change attitudes and values of people working with systems development. One alternative might be to educate people with no previous usability expertise in usability methods" (Eriksson et al., 2009, p. 551), possibly enhancing developers' qualifications in terms of UX skills (Øvad and Larsen, 2015).

Yet, "not much research has been conducted in the area of training software developers in usability and UX methods" (Øvad et al., 2015, p. 398). In fact, a relevant proportion of existing literature has rather involved students as participants (Bruun, 2010): the same author underlines how, among the 129 papers reviewed, only one (Häkli, 2005) accounts for an approach that is at the same time considerate of organisational constraints, focused on practitioners, and oriented towards user-based methods. Therefore, a need for empirical approaches that, as defined in (Øvad and Larsen, 2016b), can "leverage already existing resources in the organization – by enabling software developers to perform certain UX tasks" appears. There are several advantages to this stance, as suggested by (Bruun, 2010; Øvad et al., 2015): the potential of easing difficulties due to the lack of usability specialists in the industry, as experienced in large companies; the opportunity for small companies to lessen the need to staff usability specialists, which cannot be funded anyway; a good fit with the Agile feature of team members being able to perform every given work task.

6.2 From communication breakdowns to focal points

As mentioned in section 2.2, this thesis is positioned within qualitative research at the crossroads of CSCW and SE. The understanding practice stage, culminating in the identification of a set of communication breakdowns in the integration of UCD and Agile, relates to the interest of the CSCW community for comprehending work practices, their rationale, and related issues. In the rest of this thesis, such understanding will inform the deliberation, implementation, and evaluation of improvements of software practice as advocated by empirical software engineering instead.

As explained in sections 6.1.1, several other authors such as (Cajander et al., 2013; Damodaran, 1996; Lárusdóttir et al., 2012) have suggested that the solution to communication breakdowns may be found at the organisational structure level, advocating the creation of a supportive organisational environment where the management should actively commit to an unequivocal inclusion of UCD in company goals and financial allocation (Venturi et al., 2006), and explicitly counteracting a mindset excessively focused on technical aspects (Ardito et al., 2014; Hussain et al., 2009a). Empirical evidence supporting this interpretation also appeared in the H-umus study (section 5.3): the issues voiced by designers about the ownership of UX were related to the need for the organisation to explicitly acknowledge the dignity of UX skills rather than implicitly considering them as common sense. In agreement with this school of thought, this section will describe the second contribution of this thesis.

Communication breakdowns manifest at the work practice level: they can be compared to the visible symptoms of an illness. Let us take the example of a sore throat: one could decide to simply treat the symptoms, e.g. by drinking some milk and honey, or to perform a more detailed examination to diagnose the root cause of the illness, and then treat it accordingly, for instance by taking an appropriate antibiotic. In this sense, the goal of the second phase of this thesis is to identify and treat the root cause of communication breakdowns, or in other words to improve practice by enacting change at the organisational structure level. To be able to do so, it is useful to have a diagnostic tool that can help assess the extent of communication breakdowns in the organisation in order to collect the information needed to ground an ameliorative intervention on company policies and procedures. Contribution C2 of this thesis provides such a tool, namely a framework in which each communication breakdown is abstracted into a focal point relative to the organisational structure level. In terms of the organisational context framing described above (Bjørn and Ngwenyama, 2009), this framework of focal points translates communication breakdowns from the work practice level to the organisational structure level, as shown in Figure 14.

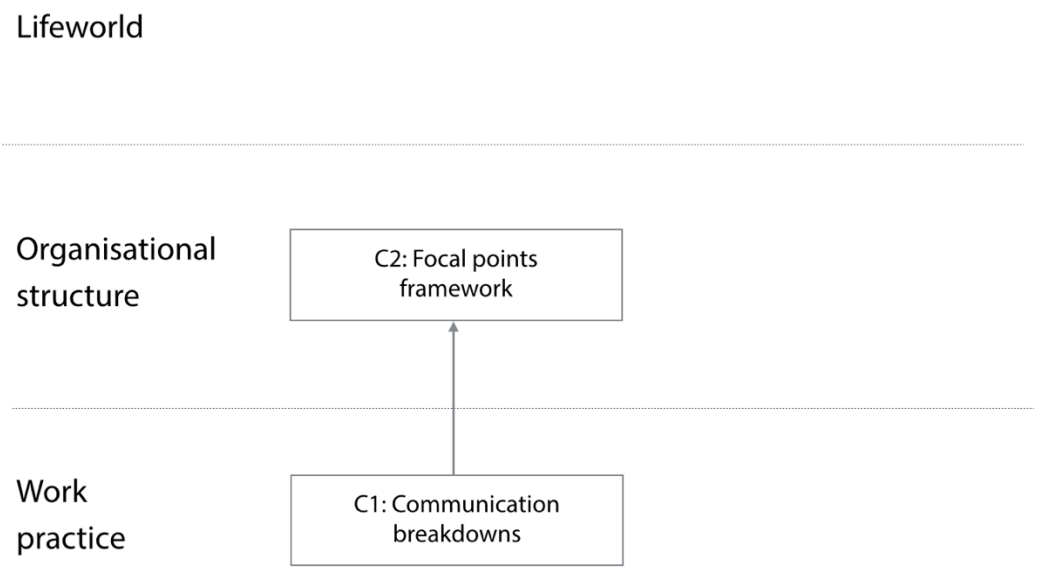


Figure 14 - from communication breakdowns to a focal points framework

Focal points are conceptual lenses for the interpretation of the status of the organisational context where the integration of UCD and Agile is to be enacted. They drive researchers' and practitioners' attention to potential communication breakdowns that are likely to emerge in this context, highlighting remarkable concepts on which an organisation should reflect to assess the extent of such breakdowns. Here follows a description of the three focal points belonging to the framework, which highlights the mapping of each of them to the corresponding communication breakdown (Figure 15).

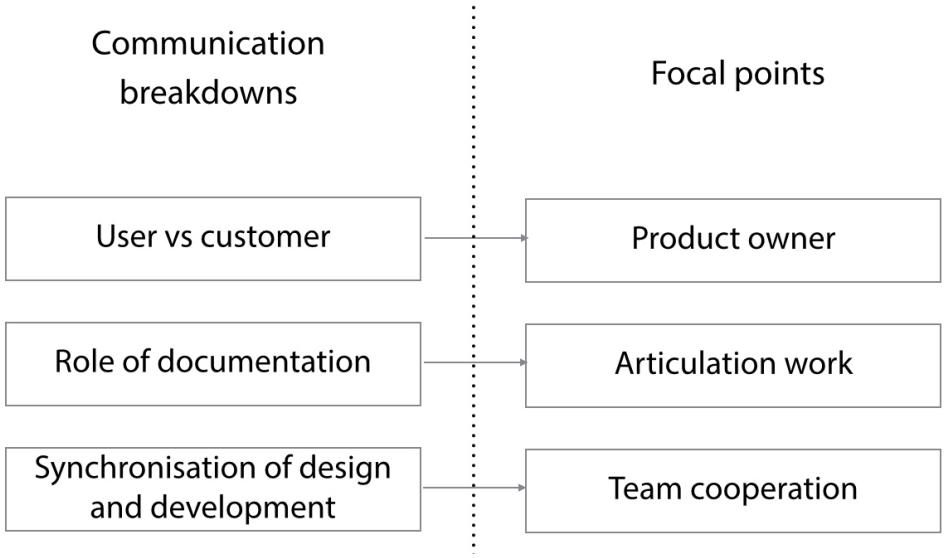


Figure 15 - mapping from communication breakdowns to focal points

Product owner. This focal point refers to whom is responsible, outside the development team, for steering the evolution of the product. Depending on the extent to which the organisation prefers to accentuate UCD or Agile in its processes, this role can be covered by one or more representatives of the user and/or the customer, who can be involved to a varied extent (see section 2.3.2.1). Even though the label of "product owner" derives from the Scrum terminology (Schwaber, 2004), this does not imply that Scrum is the only approach to Agile to which the framework is applicable; rather, its definition appropriately represents the features that I envision for this role in the framework, i.e. representing stakeholders needs, putting forward requirements and goals, prioritising tasks over time, maintaining the overarching design vision.

Articulation work. As explained in section 2.2, Strauss defines articulation work as "the specifics of putting together tasks in the service of workflow" (Strauss, 1988, p. 164). Articulation work is entailed by cooperative approaches such as UCD and Agile, and even more by their integration (Ferreira et al., 2011). This focal point refers specifically to how articulation work can be adequately supported by documentation, in terms of what should be documented, how, and why. Even though informal communication plays an important role that is particularly explicit in the Agile philosophy (Beck, 2001), the organisation may wish to establish some norms about the meaning of "enough" documentation.

Team cooperation. Schmidt defines cooperative work as occurring when multiple actors required to do some work are mutually dependent and thus need to coordinate and integrate their activities to accomplish their goal (Schmidt, 1994). Both UCD and Agile support cooperative work (Ferreira et al., 2011): this focal point asks the organisation to reflect on the way of synchronising design and development activities, but also of coordinating people engaged in this. In addition, this focal point also incorporates considerations on the "Sprint 0" (another concept borrowed from the Scrum terminology), meaning how activities, and particularly design ones, should be allocated between the upfront phase of the process and the rest of it.

As shown in Table 4, each focal point is articulated into a set of questions meant to facilitate the organisation in reflecting on several aspects of the integration of UCD and Agile, and later in making decisions about how to shape it.

Table 4 – framework of focal points to drive the integration of UCD and Agile

Focal point	Questions
Product owner	Are the user and the customer distinct? How are they characterised? How are they related? Is it possible to access a user or customer representative willing to be engaged in the system design and development? In which terms can this person be involved? Which of these user or customer representatives is the main requirements provider? With which members of the team does she communicate? (product owner, appointed developer, management…) Is one representative of users and / or customers sufficient? If not, what are the roles of significant representatives to involve? Is the customer or the user responsible for decisions about the product (including prioritisation, features, and UX)? Do they need to be supported by a team member in this role? Who is responsible for monitoring and ensuring the consistent translation of the design into implementation? To what extent are the user and the customer involved in the design and development (informative, consultative, participative role)? What kind of input and feedback is expected from them?
Articulation work	What needs to be communicated? To whom? What is the purpose of the documentation? What should be documented? How can the knowledge about the user / customer and its working context be documented? In what form(s) should documentation be recorded? How should the design vision be documented? What is the role of documentation in supporting communication within the team? And in supporting communication with the customer and the user? Who is going to be reading, writing, and maintaining the documentation? (management, developers, customer…)

Team cooperation	Should design and development activities be considered as part of the same process or should they proceed in parallel? If so, how are they related and synchronised? What kind of interaction is envisioned between team members in charge of design and development respectively? (pair designing / programming, informal vs. structured…) Which activities should be included in Sprint 0? How much user research and design can be performed upfront? How should the findings be shared within the team?

It should be noted that the potential impact of the adoption of the framework may be limited if the customer does not share the same values underlying it, for instance preventing access to users, completely mediating the communication with them, or refusing to engage in the interaction with the development team. A direct contact with users can allow the company to deliver a product that, although requiring a possibly longer design period, will be more suited to the needs of people ultimately using it and will therefore bring more value to the customer for its money. Yet, some customers and technical personnel still need to be persuaded of these advantages (*The Standish Group's CHAOS Report*, 2014) in order to win their acknowledgement of the advantages of involving the users and their active collaboration in this. To this end, however, a set of actionable measures that can more objectively assess the positive impact of user involvement on the quality of produced software (Mao et al., 2005) is still lacking, together with a set of less resource-intensive practices to put such involvement in place (Øvad and Larsen, 2016a).

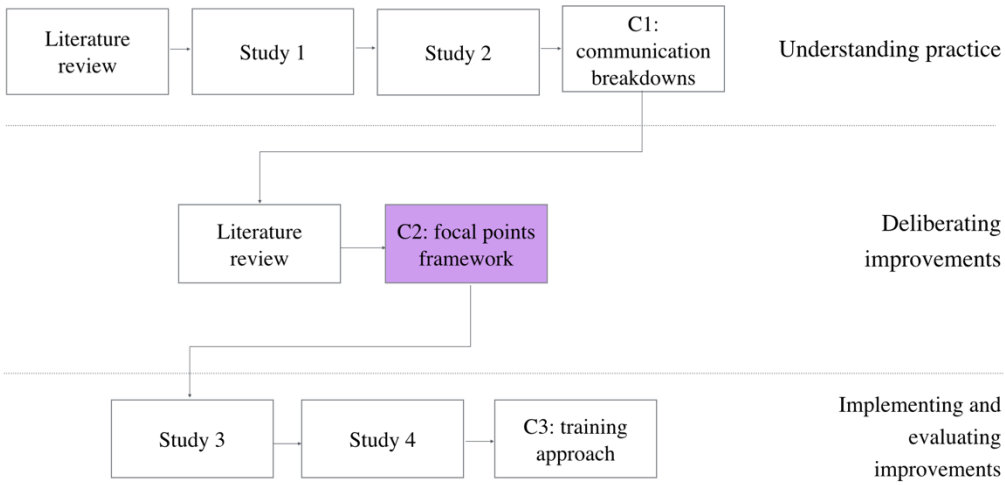


Figure 16 - dissertation progress: contribution C2

As shown in Figure 16, this section has presented the second contribution (C2) of this dissertation; these considerations provided the starting point for the third stage of this piece of research, dedicated to implementing and evaluating improvements.

6.3 Applying the framework in practice

The framework of focal points presented above as contribution C2 could have just been handed in to companies as a plain list of questions: however, this choice did not seem likely to effectively foster an improvement of practice. The goal at this point was to answer research question RQ3: "how can the practical solutions identified in RQ2 be adapted to the specific setting of small organisations?" (see section 1.2). To this end, I devised a training for companies on the adoption of the framework: in the rest of this chapter, I will introduce the training, while in Chapters 7 and 8 I will discuss two studies in which the training was applied and validated, therefore constituting contribution C3 of this thesis.

I chose to provide a training alongside the framework itself because according to constructivist theories, instructional support can provide several benefits. Since learning is an active and social process, it is particularly effective if it occurs in the zone of proximal development (Vygotskij and Cole, 1978), that is the field between what the learner can do by herself and what she can achieve with the help of a more knowledgeable instructor. The support offered by such instructor during the learning process is called scaffolding (Wood et al., 1976), and should be gradually removed as the learner becomes more proficient in mastering tasks: in this spirit, the training focused on projects familiar to participants, and included collaborative, hands-on, supervised activities meant to enable the autonomy of participants in adopting the framework and ensuring that their zone of proximal development was appropriately leveraged.

The training follows the suggestions in (Øvad and Larsen, 2016b), relying on one-day in-situ training sessions with software developers and keeping a focus on hands-on experiences and real-life cases. The training consisted of a brief series of workshops led by me in the role of a facilitator: during these workshops, developers were guided through the re-design of a relevant interface selected from an existing project of the company, applying a curated set of UCD and Agile techniques. In the rest of this section, I will just provide an overview of the principles that grounded the training: because of the importance of tailoring interventions in practice to the specific setting in which they will take place (as extensively remarked in section 2.2), the details on how the training was instantiated will be specified in Chapters 7 and 8, within the description of the two studies in which the training was performed.

As illustrated in section 6.1.2, literature presents few attempts at training developers on UX specifically. This approach, however, seemed particularly suitable in consideration of the type of companies that were accessible to me at this point of my research work: the vast majority of these companies was small-sized, as peculiar of the Italian industry. In this context, training allows leveraging existing resources. Proposed techniques include user task analyses, stakeholder mapping, personas, storyboards, wireframing and prototyping, user evaluation, and the use of a Scrum board. They were chosen according to two main criteria:

- reflecting surveys on the usability techniques most used in industry (Da Silva et al., 2015; Hussain et al., 2009b; Jia et al., 2012; Øvad and Larsen, 2015), particularly accounting for their feasibility of integration into an Agile environment and of teaching non-UX professionals;
- promoting collaborative reflection and articulation to answer the questions in the framework of focal points.

The training proposed here differs from similar attempts reported in literature primarily because of its tight relation to the framework of focal points. Furthermore, it goes beyond the scope of works such as (Bruun, 2010) and (Häkli, 2005) as it includes, but is not limited to, usability evaluation. In addition, it is indeed grounded in a similar organizational context as (Häkli, 2005), but rather than just aiming at enhancing developers' skills, it is meant to translate the framework of focal points back into practice, supporting an evolution of the organisational structure and of the company's lifeworld. Finally, with respect to Øvad's work (Øvad et al., 2015; Øvad and Larsen, 2015, 2016b, 2016a), it exposes developers of a same company to several usability methods rather than focusing on just one of them.

It should be noted that this intervention is meant for "supporting developers during ongoing day-to-day product development" (Øvad et al., 2015; Øvad and Larsen, 2015, 2016b, 2016a) and that it is not meant to replace the need for a usability expert, in agreement with (Eriksson et al., 2009). In addition, the peculiarities of each organisational context determine the applicability of the questions in the framework, and in turn the choice of techniques that may be most meaningful to developers case by case.

Referring to the analytical categories of the organisational context introduced in (Bjørn and Ngwenyama, 2009), the proposed training is positioned at the organisational structure level (Figure 17), as it aims at intervening on the methodologies in use within the company. Chapters 7 and 8 will illustrate two iterations of action research in which the core of the intervention consisted in training on the adoption of the framework, with the objective of supporting companies in shaping their own way to the integration of UCD and Agile.

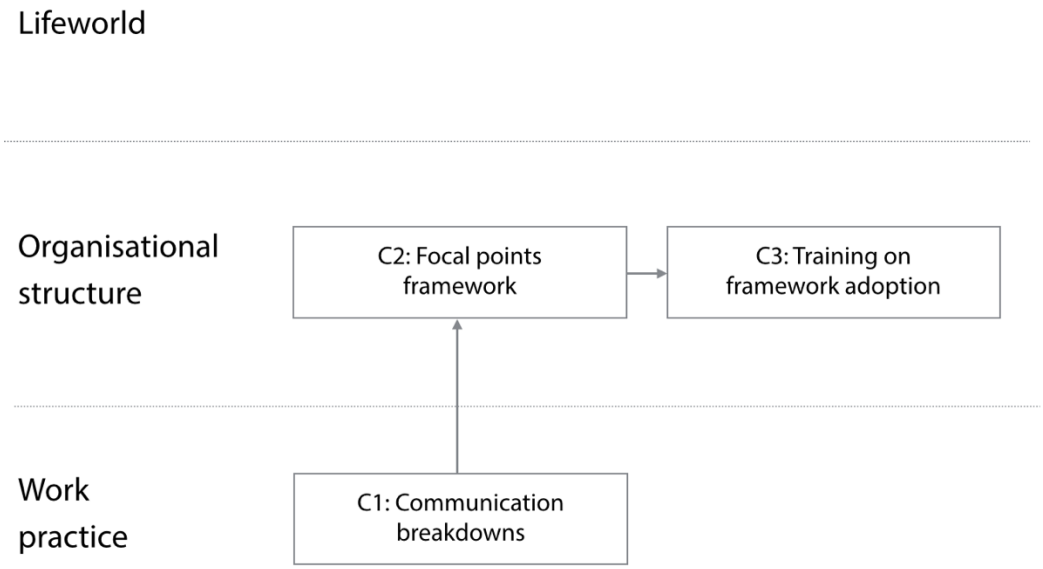


Figure 17 - from focal points to training on the adoption of the framework

7 Study 3: Delta Informatica

This chapter presents the first action research study in which a team of developers was trained on the use of the framework of focal points.

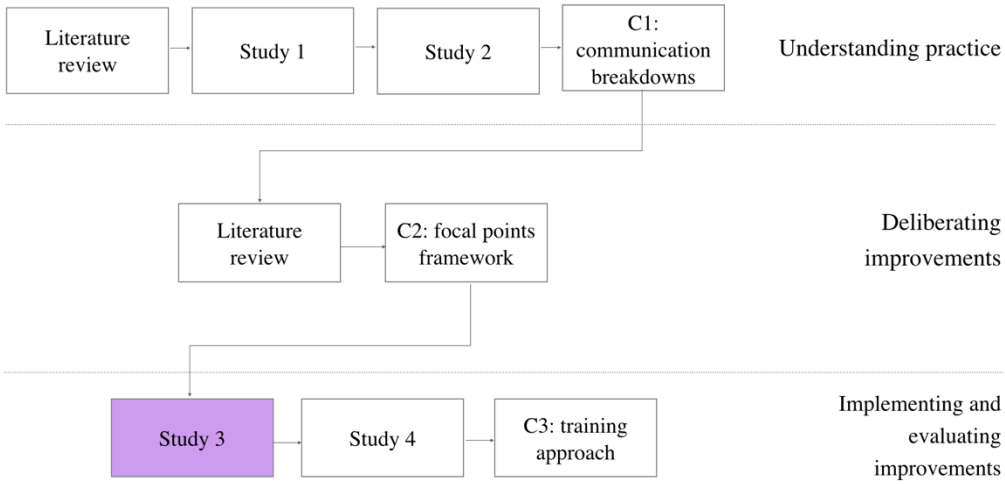


Figure 18 - dissertation progress: study 3

7.1 The context

The activities described in this chapter were carried out in the R&D department of Delta Informatica, an Italian software engineering group that provides services and consultancy mainly to SMEs or to the public administration. Within this department, I collaborated with a small development team in charge of implementing a virtual reality (VR) simulator to be used for safety training in local hospitals: this software, called PRESTO (acronym of Plausible Representation of Emergency Scenarios for Training Operations), and the development processes revolving around it have been the focus of the present study. The software was commissioned by the local healthcare agency to enhance their employees' safety training with more realistic simulations of critical incidents. A typical example of the context in which PRESTO was used is shown in Figure 19: a trainer delivers the lecture to a class of healthcare workers and is assisted by an operator, who interacts with PRESTO by managing the evolution of a VR simulation of a critical situation (e.g. a fire in the hospital). In particular, the simulation relies on a predefined "script", configured in advance by developers: the operator, similarly to a movie director, can at any time and in agreement with the trainer choose which "scenes" to activate among the plausible ones suggested by the system based on the script, allowing the plot to unfold.

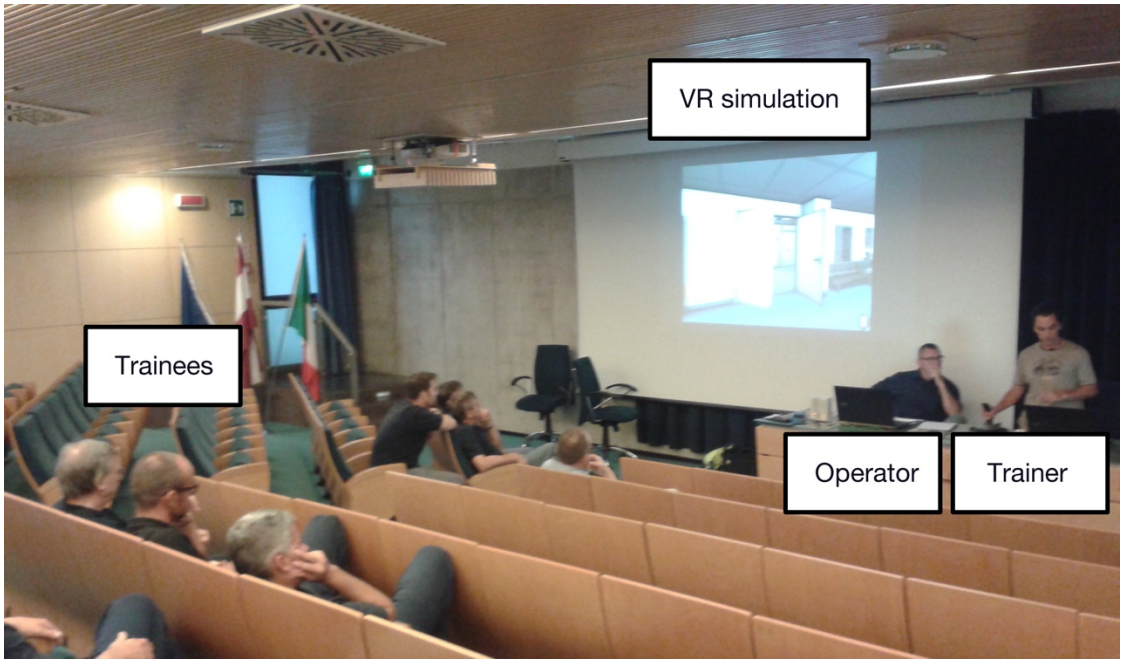


Figure 19 - Setting for the usage of PRESTO

A first version of PRESTO already existed before the intervention. However, both the Delta team and their customer were dissatisfied with it: interacting with PRESTO was in fact quite cumbersome, to the point that the project manager needed to play the role of the operator during safety classes in which the software was used. In addition, Delta did not envision to train a customer employee on the usage of PRESTO as part of their business model, so a revision of the system, and especially of its interface, was urgent. The reasons for PRESTO being so complex appeared to be mainly two. First, the environment in Delta was predominantly technical, as the team was fully composed of pure developers; none of them was formally trained in interaction design or similar disciplines, although there seemed to be a generally positive attitude towards UCD. Second, the team did not share a common vision of the project goals, hence its leader hoped that an external contribution would ease the tension between different perspectives and help the project move on.

7.2 Understanding practice

This study, as well as the one that will be discussed in Chapter 8, relied on an action research approach (section 3.4.1): specifically, I chose to follow the Cooperative Method Development (CMD) approach as explained in section 3.4.2. Activities started with a thorough, ethnographically-informed analysis of Delta in order to clarify the reasons for the status of PRESTO. This step, labelled as “understanding practice” in agreement with the CMD stages (Dittrich et al., 2008), had been the focus of the work of fellow researchers, who had already engaged with Delta before I got involved in this specific study. Fellow researchers interviewed

the team about the evolution of the software, and carried out a preliminary series of workshops to probe developers' understanding of the user they were targeting. Major misunderstandings and miscommunications among developers became apparent at this point, as will be described in the rest of this section.

7.2.1 Preliminary interview study

A preliminary interview study involved fourteen people who collaborated in the design and development of the first version of PRESTO: seven of them were proper developers in Delta, while the other seven were computer science students who participated in the development at different stages of the project during an internship. Interview guides are available in the appendix (section 10.3.1). Quotes from the interviews to developers have been coded as Dev1 – Dev7, while quotes by students have been coded as Stud1 – Stud7; all of them have been translated from Italian.

A fundamental issue concerning the identity of the intended user of PRESTO emerged soon. In fact, the definition of such user was clearly not shared:

Dev1: "In the project we envision [the local health authority] as the first user who is collaborating in the project"

Dev3: "The final user in PRESTO is the trainer […] I believe he has been the target user since the beginning of the project"

Different opinions on the nature of the user, and subsequently on the nature of PRESTO, led to conflicts between the project manager and the team leader: the former considered the system as an open-world videogame, or a thin gaming layer over the complex artificial intelligence engine that actually handled the evolution of the virtual reality scenario; the latter instead understood the system as an end-user development tool that a non-trained operator could use to dynamically configure and play a new script based on the requests of the safety training class. There did not seem to be an agreement even on whether the end user should be one or many:

Dev6: "We actually have several end users and in the end we realised it would be better to focus on the main one, the trainer"

Dev7: "We now have two users. We took the trainer […] and we split it into two users: one […] called training strategist, who defines the training goals, defines the simulations scenarios […] and then we have the trainer, who takes one of these fully-configured scenarios and may customise it to a limited extent"

The so-called "training strategist" is a particularly interesting figure: as a matter of fact, no such user exists among the personnel of the local health authority meant to adopt the software. It appeared that this role had been made up by developers in response to the growing complexity of the system:

Stud4: "So this new figure called training strategist is located between a developer and a trainer […] This re-definition of the user was due to continuous changes that contributed to making the system increasingly complex"

The growing intricacy of the software was not however the only reason for the ideation of the training strategist. In fact, the whole team shared a concern about the difficulties experienced in accessing real users with whom to check the software evolution:

Dev1: "For us it is important to know the requirements of our final users. To date, interactions with what could be the first final users have been few. Therefore, since we had little feedback, we tried to imagine […] what could be the requirements and the user"

Dev7: "There was the issue of the existence of few users, to whom we could not get access"

Dev6: "It has not been easy to find users, plus there have been issues with the few we could get in touch with, so it has not been possible […] to keep them involved in the design process"

Nevertheless, the team generally acknowledged the relevance of user involvement in the project, although opinions on its extent varied:

Stud6: "[The user] should be present at design time and at implementation time as well"

Stud7: "[The user should have] mostly the role of a feedback provider"

Dev4: "In my opinion it is better to lay the foundations of the software in the first stages, but then it is definitely very useful to get user feedback"

Stud1: "The final user […] should have a role in interface development, but definitely not in software design"

In summary, even though the development team had a generally positive attitude towards embracing users' needs and feedback, at this point no shared understanding of the user identity and of the extent of its involvement appeared. These underlying inconsistencies were further highlighted when a lighter version of PRESTO was used during an academic master course in artificial intelligence: despite an introductory training, students repeatedly reported difficulties in the interaction and misunderstandings in the interpretation of the system semantics.

7.2.2 Preliminary workshops

To consolidate the understanding developed in preliminary interviews, three workshops were organised in summer 2015 with the development team, and were led by fellow researchers. These workshops included activities such as empathy maps (Gray et al., 2010), two-level personas (Dittmar and Hensch, 2015), and cognitive walkthroughs to elicit the mental model of the user that developers had consolidated over time: such model was partially based on the experience in safety classes reported by the project manager, but largely relied on developers' own common sense. Additionally, an ethnographic observation of a safety training class held at the local healthcare institution was performed: this was followed by interviews with a few key participants, namely the trainer, the developer in charge of configuring PRESTO for the lecture, and the operator who actually interacted with the software during the class. Insights from these interviews supplemented the outcomes of preliminary workshops, particularly highlighting the need to consider the operator as a fourth kind of potential user of PRESTO.

Fellow researchers concluded this preliminary stage of the study with a report that identified four categories of potential users of PRESTO which emerged from the findings of preliminary interviews and workshops and whose main traits can be summarised as follows:

- **trainees**: healthcare professionals who need to be trained on safety. They are not computer science experts; they interact with the system indirectly, in the sense that they discuss with the trainer about what kind of interventions they would suggest in the simulation running.
- **trainers:** safety professionals delivering the course. They also do not interact directly with the system, but rather give instructions to an operator about how the virtual reality scenario should evolve.
- **training strategists:** skilled computer scientists, knowledgeable about artificial intelligence, and capable to create, modify, and use virtual reality scenarios. As mentioned above, these users have been envisioned as a response to the lack of access to proper users and to the growing complexity of the system.
- **operators:** the only ones interacting directly with the system: they are usually "casual" users, asked to play this role occasionally and on the spot, and may have a variable level of computing skills. Interestingly, the existence of these users only became apparent during the ethnographic observation.

At this point, I took over the study. Following the framing described in (McKay and Marshall, 2001) and section 3.4, I specified the company's problem solving interest and my own research interest. The former concerned how to enact a development process that suited Delta's own

contingencies of being a small enterprise, retaining the benefits of both UCD and Agile in terms of focus on the user and of rapid, flexible implementation respectively; the latter regarded assessing if training developers would be an appropriate way to facilitate their adoption of the framework of focal points, and subsequently whether existing issues could be mitigated by this intervention.

7.3 Implementing improvements: design workshops

The stage of understanding practice (intended as the first stage of the CMD approach, and described in section 7.2) highlighted major discrepancies in the perceptions of users exhibited by participants. Concurrently, around December 2015, Delta committed to deliver a redesign of PRESTO by March 2016, and wished to structure a working process that would allow the team to meet the deadline while producing an improved interface. In the CMD terminology (Dittrich et al., 2008), this allowed to move to a phase of improvements deliberation, i.e. to train the team on the use of the framework of focal points (Chapter 6). The intervention was designed based on the focal points listed in the framework; improvements were initially deliberated in collaboration with the team leader, and then cooperatively implemented in a series of workshops with the team. Because the team did not follow a specific design or development approach, the training was articulated in two parts, each consisting of a set of workshops: the first part focused on design (described below), the second part focused on development (described in section 7.4). The bases underlying the training have been covered in section 6.3. The specific activities proposed to participants were partially chosen in advance, based on the weaknesses of the team and of the project spotted in the understanding practice stage of this study, and partially selected or adjusted along the way based on the outcomes of each workshop. Five design workshops were scheduled with Delta's development team; Table 5 provides a more detailed overview of their timing and duration, while their agenda is outlined in Figure 20. They were held approximately every two weeks at the University, thus providing a neutral environment for participants, and were led by my advisor and myself together.

Table 5 – timing of design workshops in Delta

Workshop	Date	N. of participants	Duration
W1: product owner identification	22/01/16	5	2 hours
W2: design space	29/01/16	5	3 hours
W3: non-functional requirements	05/02/16	5	2 hours
W4: terminology	11/02/16	6	2.5 hours
W5: design completion	17/02/16	4	2 hours

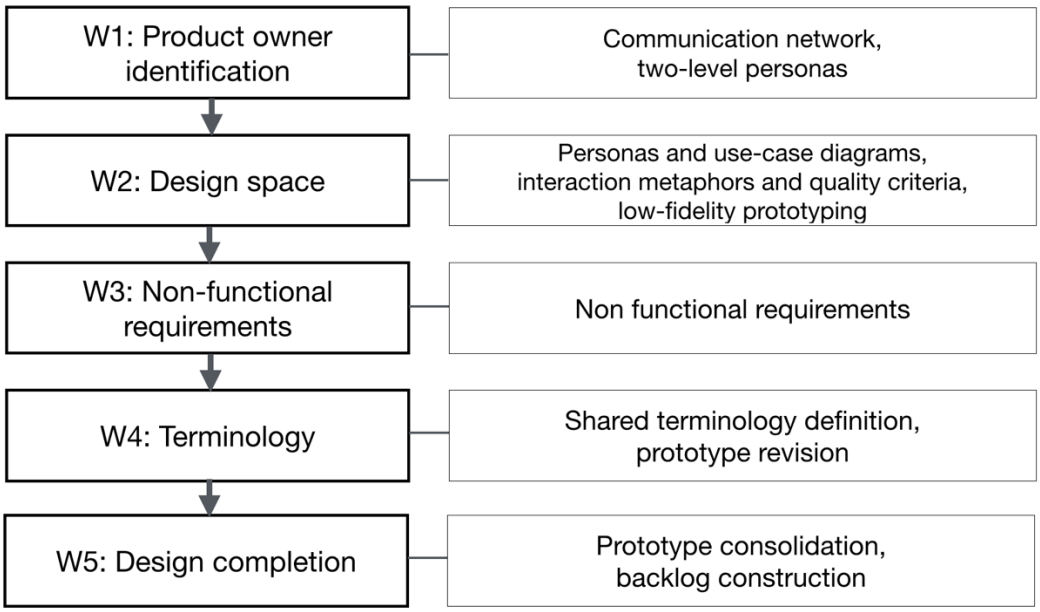


Figure 20 - overview of the design workshops

The structure of the workshops adjusted over time based on the outcomes of each workshop and on participants' interests, following the guidelines of the CMD approach on deliberating improvements together with the involved practitioners (Dittrich et al., 2008). Design workshops can be interpreted in reference to the framework as a guided "Sprint 0", where collaboration and information sharing are established within the team. Three developers (one specialised in the front-end, one in the back-end, and the project manager) and one intern attended the whole workshop cycle; occasionally, a couple of other colleagues and the team leader also joined. Except for the intern, who had attended an HCI course as part of her academic education, the other participants did not have any formal training in UX or in UCD approaches. The front-end developer had a personal interest in UX themes, but pursued it autonomously and informally.

To protect confidentiality, relevant quotations from the participants will be coded anonymously as D1 to D5 in the following.

7.3.1 Workshop 1 – product owner identification

While the identity of the customer of PRESTO was clear, a major issue concerned establishing the identity of the user intended for PRESTO, based on the limited information available, and what kind of feedback could be expected from such user or its representatives. Therefore, this first workshop aimed at defining the characteristics of the product owner: we suggested taking one step back and visually representing the communication network (Cataldo and Herbsleb, 2008) of actors and artefacts revolving around the software. This activity turned out to be extremely hard: after an hour of heated debate, participants had not been able to converge on a representation of the stakeholders involved in the project, nor on identifying the most relevant subset of them. In addition, the divergence in the visions of D1 and D2 did not foster a constructive dialogue.

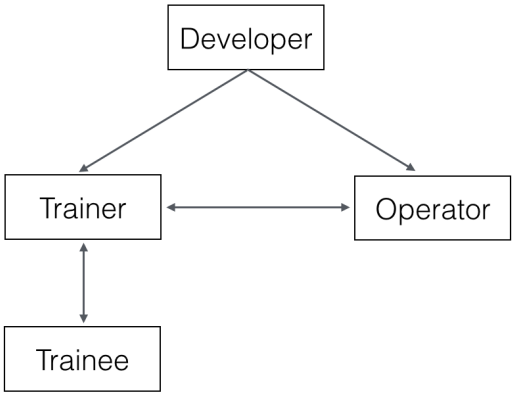


Figure 21 - Reference communication network for PRESTO

To help the discussion move forward, we established what seemed the most meaningful subset of the communication network, illustrated in Figure 21. Such subset shows how PRESTO developers interact both with trainers and operators to configure the software and set up scripts for specific lectures; moreover, it also explains that trainees discuss with their trainer during classes and that the operator coordinates intensely with the trainer for the purposes of delivering an effective lecture. Participants were asked to collectively elaborate the profiles occurring in Figure 21, eliciting their intended characteristics; as the discussion progressed, emerging user features were noted on a large whiteboard, in order to provide a shared visualisation. It soon became clear, however, that most information was contributed by D2, the only workshop participant who had actually attended safety classes where PRESTO was in use and who had

spoken with trainers and operators. Both domain and user knowledge were therefore mediated by him and by his strong personal interest for videogaming.

Participants were finally asked to collaboratively build two-level personas (Dittmar and Hensch, 2015) starting from the point of view of the trainee and from his ultimate goal of learning how to deal with dangerous situations: they had to describe how pursuing this goal could affect the relationship of the trainee with the trainer and with the operator, and in turn with the developer. We suggested exploring several alternatives, differing in the degree of technical and domain skills that the trainer and the operator could have. In comparison with the two-level personas mentioned in section 7.2.2, participants at this point had elaborated a more grounded reasoning on the features of the actors related to PRESTO, hence we were expecting an improvement in the quality of the artefacts. Participants committed to returning the outcomes during workshop 2.

7.3.2 Workshop 2 – design space

Rather than producing proper two-level personas based on the communication network in Figure 21, participants had elaborated a short report on the mutual relationships between trainers, trainees and operators; in other words, they investigated how each of the three figures interacted with the other two during class or while preparing the lecture. In addition, they had elaborated several personas (Cooper, 2004), exploring different characteristics of each actor as follows:

- Lucia Oss: a trainee with low intrinsic motivation in attending the safety classes, but obliged to do so being the safety manager in charge;
- Fabio Trentini: a professional trainer with high technical skills;
- Mario Corradi: a trainer with less interest and familiarity with technology, yet with good domain knowledge;
- Daniele Fontanari: an occasional operator, with good technical and domain skills;
- Carlo Furlan: also an occasional operator, with good domain knowledge and a strong motivation in helping out during the course, but with less technical skills.

This helped in bringing forward further elements about the operator, an actor whose relevance had only emerged during the ethnographic observation performed among preliminary activities. Participants in fact envisioned a tight collaboration between the operator and the trainer, who agreed on the structure of the lecture before meeting their class, and a large autonomy of the operator in dynamically adjusting the evolution of the VR script based on the questions and on the level of interest of the class.

At this point, potential users of PRESTO were more thoroughly characterised, beginning to address the focal point in the framework related to the product owner: yet, given the persisting difficulty of the group in converging on a reference user and the enduring clashes in the visions of D1 and D2, we chose to further consolidate the reflection on the characteristics of each persona by formalising a task analysis activity through the creation of use-case diagrams. In particular, we aimed at eliciting the tasks that each persona was meant to perform during safety classes at broad, without referring specifically to PRESTO. Soon, participants acknowledged that the operator was the only actor interacting directly with PRESTO at the current status of the system, and was therefore the most plausible end user for the software. Figure 22 represents the related use-case diagram.

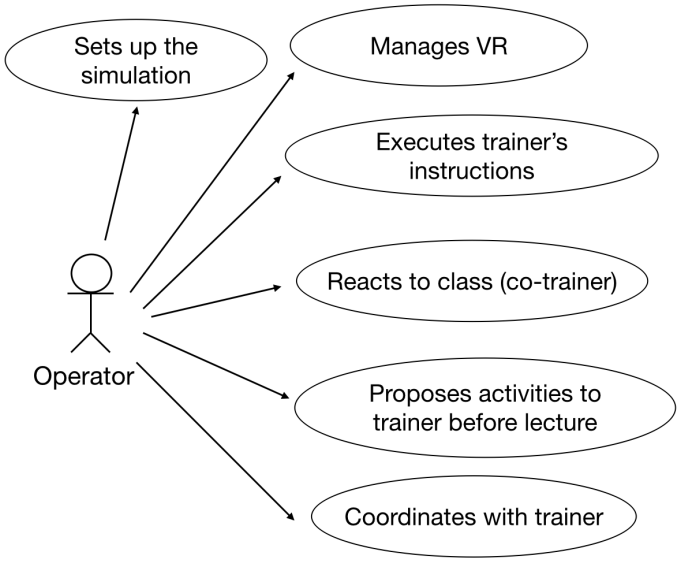


Figure 22 - use-case diagram for the PRESTO operator

In addition, participants chose as a reference user "Carlo Furlan", the persona representing the non-technically-skilled operator, over "Daniele Fontanari", the geekier one, as they agreed Carlo would be more likely to resemble the real user for PRESTO based on evidence from the preliminary ethnographic observation and as reported by the project manager based on his own experience in attending safety classes. Referring to the framework, the persona became a means to document available knowledge about the user, hence supporting articulation work within the team and addressing the focal point of the same name in the framework. I remark here that PRESTO only exposes one user interface, the one with which the operator was meant to interact; I will refer to this specific interface throughout.

Once a convergence on the reference user was finally achieved, we introduced the concept of interaction metaphor as a mediator between the system and the user, pointing out the

importance of exposing actions that could be meaningful for the user over exhibiting the complexity of the system. To clarify this abstract concept, we suggested some examples of interaction metaphor that could be adopted in PRESTO and that represented different instances of the compromise between the expressiveness of the system and the abstraction from its complexity:

- remote control: a separate controller (possibly a mobile device) that only allowed choosing among a finite set of actions depending on the context of the current scene; plot unfolding would therefore be sequential;
- interactive book: at each scene, the operator would choose among a finite set of actions, but had more visual cues available about the context; plot unfolding would still be sequential;
- Prezi-like interaction: the operator interacted with a dedicated interface that represented all possible evolutions of the story, showing a graph of enabled scenes and their transitions; plot unfolding would therefore not be necessarily sequential anymore, yet the interface required more technical skills to be understood;
- switchboard: the operator could turn on and off several switches to act on the scene, leveraging the full expressiveness allowed by the VR engine, yet not having an explicit representation of the story thread and needing a high cognitive effort to ensure that the scene remained plausible (e.g. when starting a fire, the operator should also remember to switch on the smoke). It is worth noting that this metaphor was the one used in PRESTO before our intervention.

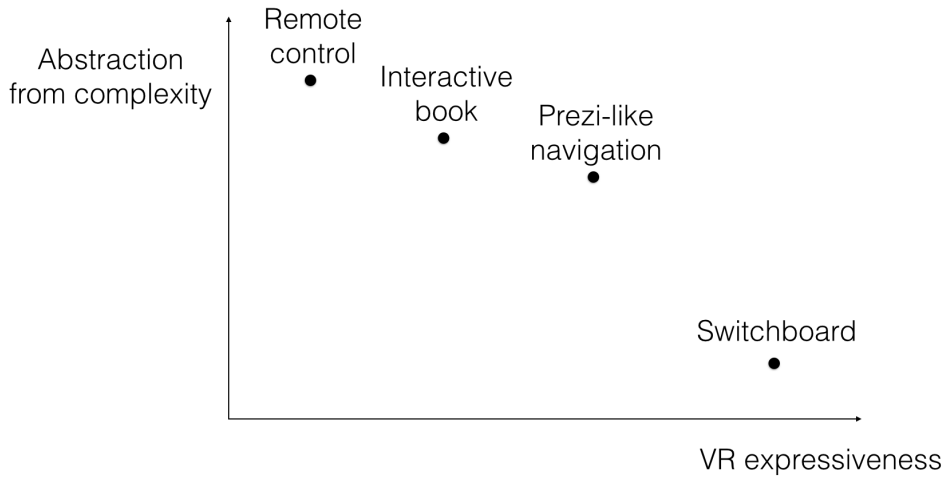


Figure 23 - alternative interaction metaphors for PRESTO

Such proposals are laid out in the diagram in Figure 23, which plots them in terms of their trade-off between system expressiveness and abstraction from its complexity. Our choice of suggesting interaction metaphors, rather than having participants freely come up with them, is due to the difficulties shown by the Delta team in devising a coherent interface for PRESTO and a shared rationale for it.

We then moved to the definition of quality criteria for the evaluation of proposed interaction metaphors. Given the highly technical background of our participants, we proposed to focus on the meaning of quality for our intended user Carlo, rather than also accounting for more functional criteria (e.g. development cost, response time etc.): therefore, we listed the five dimensions of usability proposed by Nielsen (Nielsen, 1994) and we asked participants to rate how relevant each of them would be to Carlo by applying the planning poker technique (Grenning, 2002) borrowed from the Extreme Programming approach (Beck, 2000). Table 6 summarises the outcome of this session, where efficiency, memorability, and robustness to errors were identified as the reference quality criteria for PRESTO.

Table 6 – quality criteria for the interface of PRESTO

Quality dimension	Participants' estimates (1 to 5)	Agreed priority	Reference criteria?
Learnability	5, 5, 5, 2, 4	3	No
Efficiency	3, 4, 3, 4, 4	4	Yes
Memorability	4, 4, 5, 4, 3	4	Yes
Errors	5, 3, 3, 5, 5	4	Yes
Satisfaction	4, 1, 2, 3, 3	2	No

Finally, since participants were already familiar with low-fidelity prototyping tools such as Balsamiq Mockups (https://balsamiq.com/products/mockups/), we asked them to collaboratively sketch the interfaces they envisioned for Carlo before the following workshop; moreover, we asked them to draft a list of requirements for the new interface, taking into account the elaborated persona, the chosen quality criteria, and the suggested interaction metaphors, to converge on one interaction metaphor based on the agreed priorities of quality criteria.

7.3.3 Workshop 3 – non functional requirements

Before the third workshop, we were informed that, due to imminent deadlines pertaining to other projects, workshops participants would only be able to work on low-fidelity prototypes individually; furthermore, D1 circulated a table comparing the four interaction metaphors discussed at the previous workshop that unfortunately still reflected an openly critical attitude towards D2's proposals, and did not include any reference to the user perspective. It was clear that the "developer mindset" (Ardito et al., 2014) was still strong. Therefore, although the other participants had indeed delivered a few prototypes, commenting and improving them over an email thread, we chose to reinforce user understanding by dedicating a workshop to the identification of non-functional requirements for each proposed interaction metaphor, considering in particular the quality criteria agreed in the previous workshop.

7.3.4 Workshop 4 – Terminology

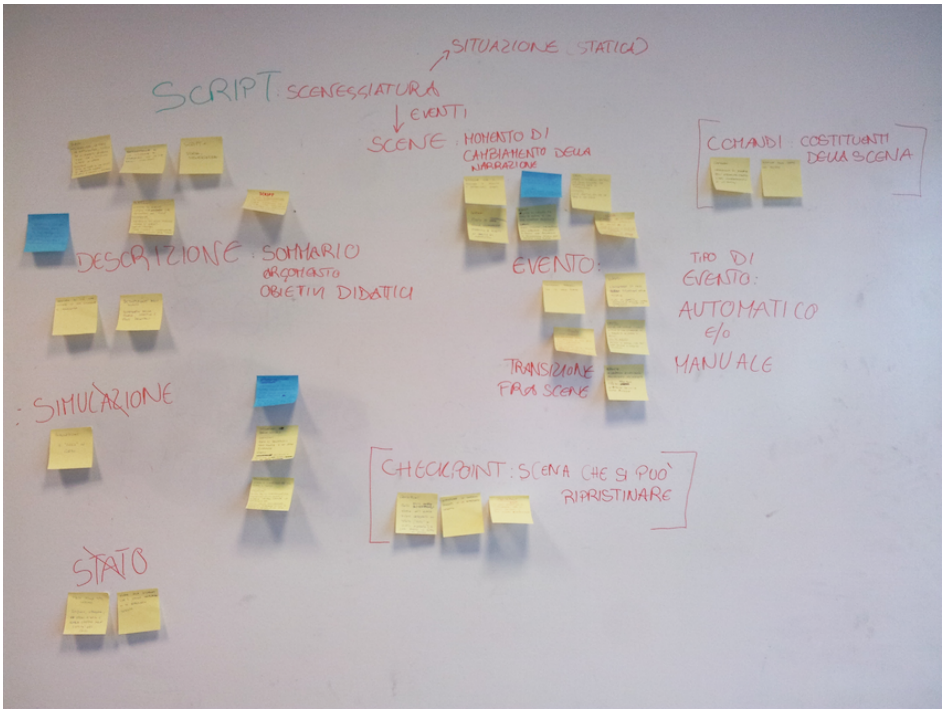


Figure 24 - construction of a shared terminology for PRESTO

Before the fourth workshop, participants delivered a list of requirements in which they made a first attempt at expressing the point of view of the user; in other words, a backlog of user storie (Cohn, 2004) started shaping. Reading such list, however, it became apparent that some terms had an ambiguous meaning and were used by different people to refer to slightly different concepts: therefore, during the fourth workshop we asked participants to individually write their definition of each notable term extracted from that list. Definitions were then stuck to the

whiteboard (Figure 24), again for the purpose of shared visualisation, and different interpretations were reconciled into a shared terminology; special care was devoted to translating technical terms, which still reflected the complexity of the underlying system, into ones more understandable for the user. This activity helped in bringing to the surface underlying conceptual inconsistencies and in harmonising residual differences in the understandings of the goals of the PRESTO system.

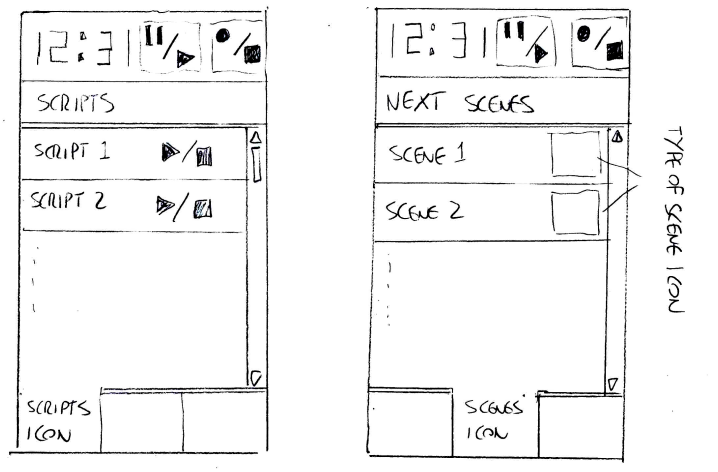


Figure 25 - hand-drawn remote control prototype

Figure 26 - remote control prototype created digitally with Balsamiq Mockups

At this point we went back to the prototypes delivered by participants for workshop 3, in order to collaboratively review them. Each participant had prepared one of these prototypes either by hand-drawing or with Balsamiq Mockups, a digital wireframing tool; as mentioned earlier, there had been some rounds of peer review among participants. Figure 25 and Figure 26 show two examples of mockups illustrating the remote control interaction metaphor: the main content of the interface is the list of possible scenes that can be activated at a certain point of the evolution of the script.

Participants pointed out that they had been reflecting on the interaction metaphors proposed during workshop 2: if the switchboard had proven unsuitable, the remote control seemed the easiest and quickest to implement, ruling out both the Prezi-like interface and the interactive book. However, participants had also discussed on the applicable extent of information hiding, i.e. on the degree of awareness with respect to the unfolding of the story that the operator needed to have in order to effectively interact with the system. They argued that the remote control metaphor was too constraining with respect to the potential expressiveness of PRESTO: therefore, they had come up with a new metaphor, inspired by video editing tools. In other words, they envisioned Carlo, the operator, as a movie director who needed to activate different story lines, possibly at the same time, and needed to know which scenes could be coming up next. Figure 27 illustrates the prototype proposed by participants themselves, which is composed of three main areas: a timeline showing the scenes currently running; a graph illustrating the next possible scenes for each running script; an area for details about a selected scene, meant to assist the operator in figuring out its content before activating it.

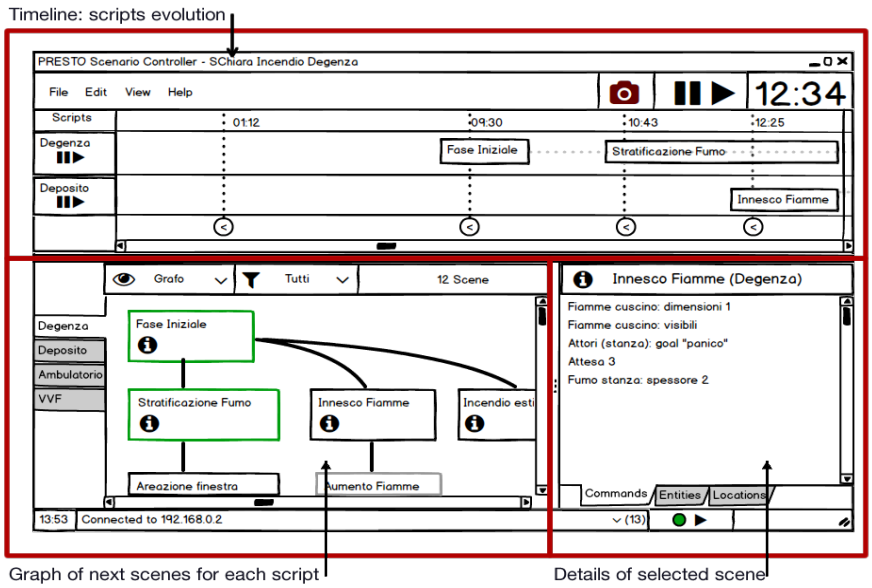


Figure 27 - video editing inspired prototype for PRESTO

7.3.5 Workshop 5 – design completion

The fresh proposal of the video editing interaction metaphor finally managed to reconcile the different visions about PRESTO: it came from the developers themselves, so it was not perceived as a top-down solution; it demonstrated an evolution in the understanding of now consolidated user characteristics; and it greatly simplified the interaction with the system. Participants explicitly asked for our collaboration in revising this first prototype, showing a strong interest in learning more about interface design and usability guidelines.

A few days later, D2 sent us an email saying that, in agreement with his colleagues, he had tried to bring the exercise further. In fact, he attached a spontaneous iteration of the prototype, resulting from a conversation he managed to have with a healthcare employee who sometimes played the role of the operator in safety training classes. The prototype had now evolved into an interface essentially divided in two (Figure 28): the upper part showing next possible scenes, and the lower part showing a timeline illustrating scenes and scripts running at a certain point in time.

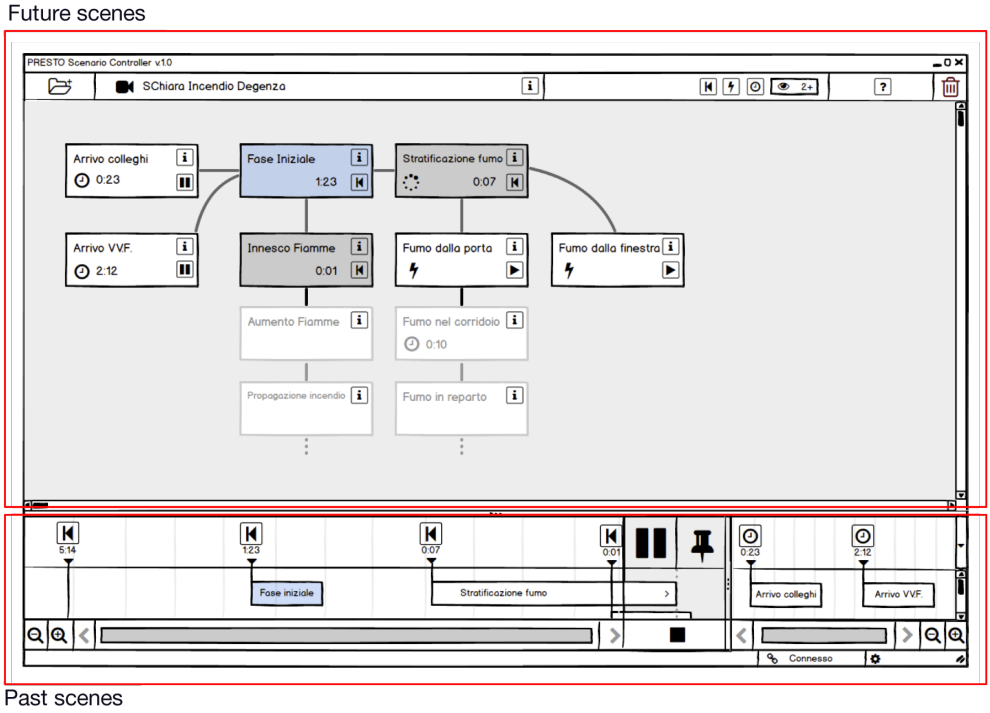


Figure 28 - prototype iteration based on user feedback

7.4 Implementing improvements: development workshops

The prototype in Figure 28 represented a pretty much stable version of the interface: we could now address improvements in the development. To this end, we planned another set of periodic meetings inspired by the sprint reviews envisioned in the Scrum approach. Up to that moment, the team did not follow a specific development process, although D2 seemed to be inclined to Agile:

D2: "I have always tried to work in an agile fashion, although not with the final customer"

Table 7 provides an overview of these meetings and of their duration. They were facilitated by myself, were held by the Delta site, and were attended by D2, D3, and D4.

Table 7 – timing of development workshops in Delta

Meeting	Date	Duration
M1: Backlog creation	25/02/16	2 hours
M2: Scrum board introduction	04/03/16	3 hours
M3: sprint review	10/03/16	2 hours
M4: sprint review	31/03/16	2 hours

7.4.1 Meeting 1 – backlog creation

The first meeting focused on the creation of a backlog: together, we went through the requirements, assessing whether each of them could be regarded as a user story or needed further elaboration. A minimal subset of these stories was selected to be addressed in the first release of PRESTO, scheduled for three weeks later. In a couple of days, participants confirmed the agreed backlog via email and attached a further collective and interactive elaboration of the interface prototype, commenting that they had also incorporated some considerations on its technical feasibility (Figure 29); for instance, scene graphs were considered too time-consuming to implement and were replaced with plain lists. It is worth remembering that the back-end of PRESTO was already in place, while the prototyped interface still needed to be implemented.

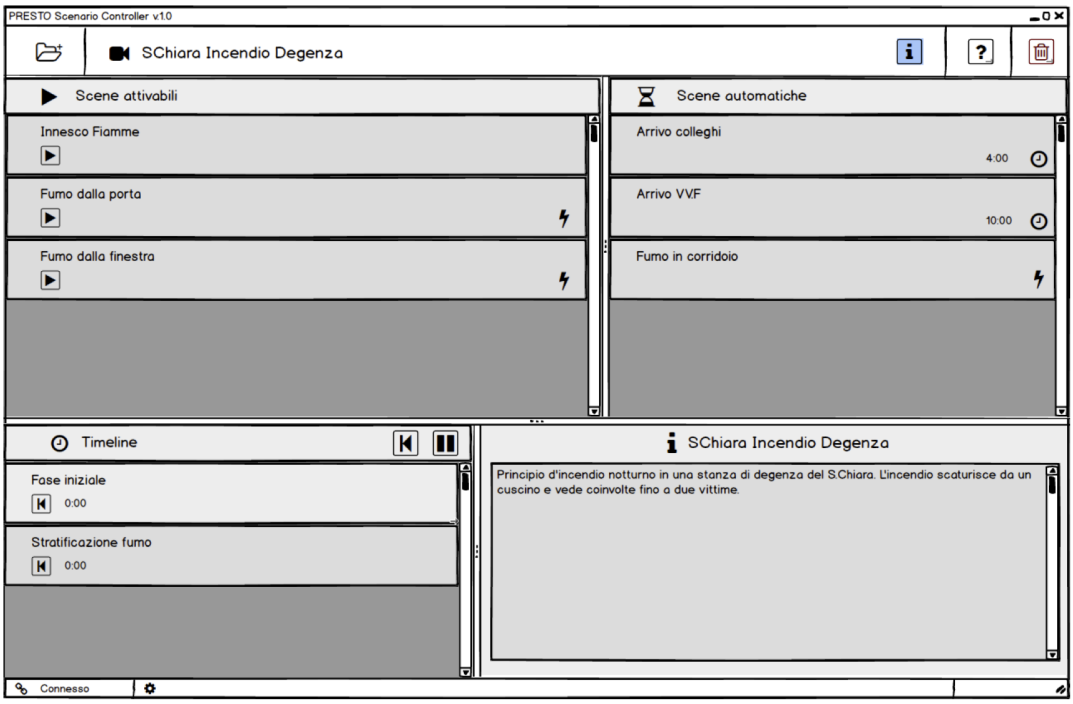


Figure 29 - finalised prototype for PRESTO

7.4.2 Meeting 2 – Scrum board introduction

Figure 30 - Scrum board with prints of the reference user persona and of the finalised prototype on top.

The second meeting focused on the introduction of Lean and Agile techniques to encourage a fast development of the interface while ensuring shared awareness of project status and constant tracking of its progress. To this end, a Scrum board was set up in the office where D2 and D4 were sitting (Figure 30). User stories were copied on sticky notes and colour-coded according to whether they referred to front-end development, back-end modification, or user testing.

Developers proposed to stick to the whiteboard also a print of the "Carlo Furlan" persona and of the interface prototype as it evolved:

D2: "I think it is better for us to have all of the materials well visible in the same place. It reminds us of the path we followed up to here."

In reference to the framework, personas and prototypes had become forms of documentation of the design vision, meant to support collaboration among developers. It is worth noting that developers immediately rejected the option of a digital Scrum board. D3 in fact sat in a different office on a different floor of the building, so the team, although small, was not entirely co-located. He stated:

D3: "I'd rather go for the physical board. If you write me on Skype asking to do something… it is different for me to say, "yes" on Skype or in person. I am much more committed if I say, "yes" in front of you. Also in this way we are forced to meet every day. I have only joined the company recently and sometimes I feel like I am bothering people when I go into their office."

Each task was also annotated with a tentative assignment to one of the developers, referring to the framework in the definition of responsibility especially for UX tasks, and an estimate of the time required to complete it. Developers were not used to providing such estimates: therefore, for each task they discussed how long it would take, and then the developer in charge of the task would make the final estimate. The backlog for the first release was split in two sprints, each lasting one week. It was agreed that the team would keep quick face-to-face daily meetings to check progress:

D2: "So you need to tell in front of everybody what you have been doing up to now and you are responsible for that"

7.4.3 Meetings 3 and 4 – sprint reviews

Before the next meeting, D3 circulated a high-fidelity prototype that further refined the interface design (Figure 31), stating that this would be the ultimate version of it and that interface design was to be considered frozen from that moment on.

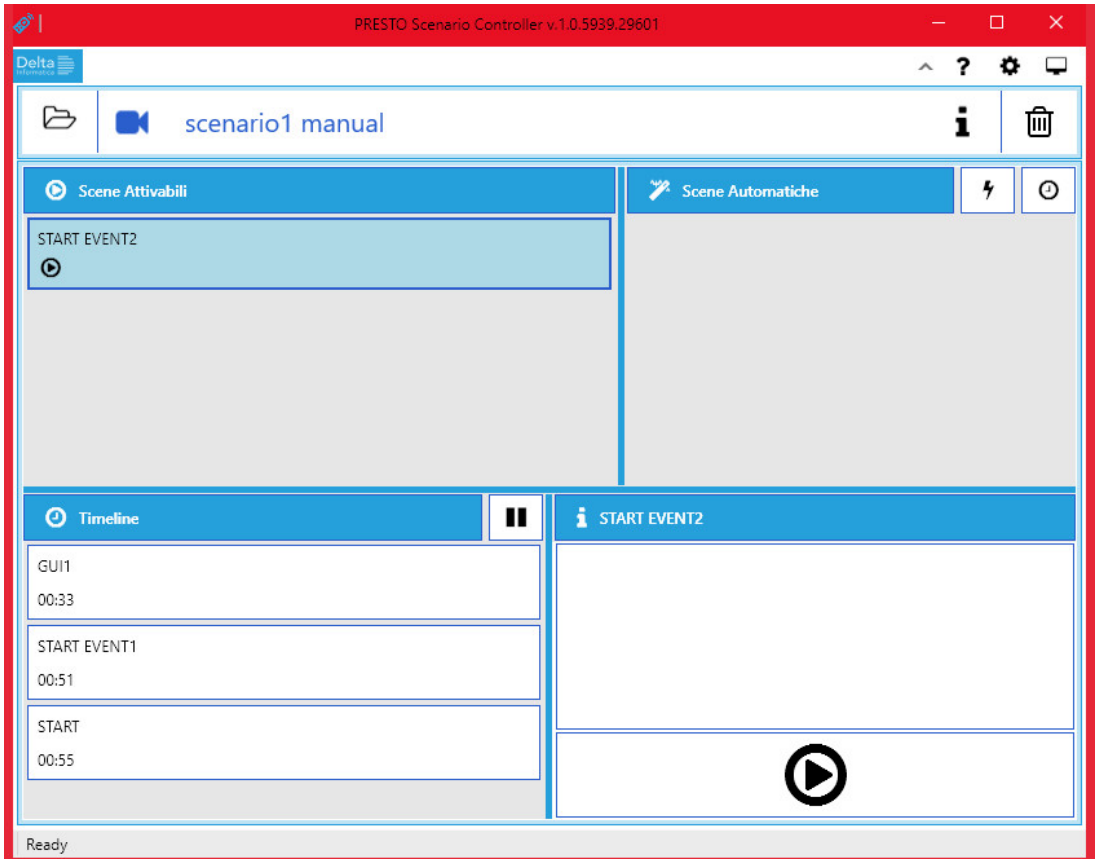


Figure 31 - high-fidelity prototype for PRESTO

Meetings 3 and 4 unfolded as sprint reviews, assessing how much of the planned work had actually been completed, and checking whether any unforeseen issues had arisen. As expected, estimates turned out to be excessively optimistic in meeting 3, and more realistic in meeting 4; however, all major tasks were completed in time for the first release of the software. After this point, we let the team proceed on their own, also because they had to complete other projects besides PRESTO.

7.5 Evaluation of improvement

The evaluation of implemented improvements addressed the problem solving interest of Delta by ascertaining whether the resulting interface was more usable (section 7.5.1), and addressed my research interest by assessing whether the training had been successful in supporting the

company in the integration of UCD and Agile and in causing a shift towards a more user-centred mindset (section 7.5.2).

7.5.1 Problem solving interest: user evaluation

In May 2016, the customer (i.e. the local healthcare agency) insisted to deploy the revised version of PRESTO during the safety training classes scheduled for that period. As said earlier, the customer was as dissatisfied as Delta with the usability of the software: therefore, Delta was keen on proving an improvement of the interface, hence we collaborated with them in structuring a user testing session.

We chose to resort to surrogate users, as access to real users was hard to achieve especially in such a short time: we thus recruited 21 master students within our department who were complete novices in the use of PRESTO. I acknowledge that our surrogate users did not fully reflect the characteristics of reference users like Carlo: however, their scarce familiarity with PRESTO and their limited domain knowledge could at least partially make up for their stronger technical skills. Students were introduced to the tool with an hour of training (reflecting the fact that learnability had not been considered as a priority quality criteria). Then, they were asked to fill in a short questionnaire inspired by the Interface Quality Scale (De Angeli et al., 2009) to assess whether the interface was usable at a macroscopic level.

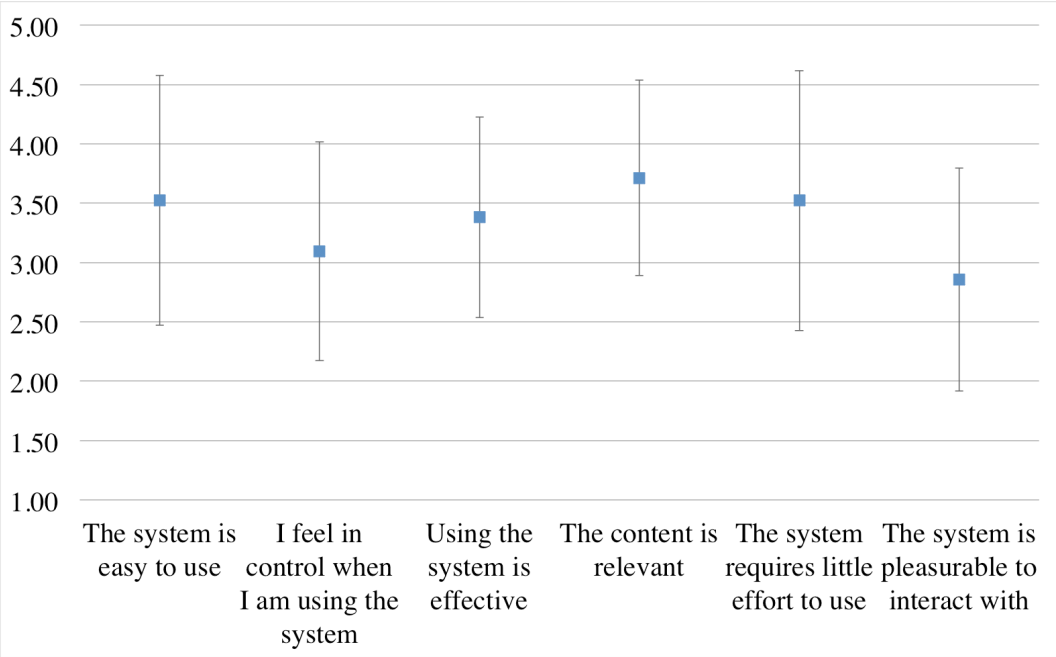


Figure 32 - outcomes of user testing

The results of this user evaluation indicated that the interface had reached a good level of usability (Figure 32). Most interestingly, answers reflected the quality criteria for the interface agreed during workshop 2 (section 7.3.2): for instance, satisfaction in use had not been considered as a priority.

7.5.2 Research interest: interviews

The evaluation of the impact of the training in Delta was carried out in June 2016. Three researchers who had not taken part in the intervention interviewed the five workshop participants, in order to avoid social bias. They enquired on whether they found the training (and implicitly the framework) useful in supporting the team in shaping their own integration of UCD and Agile, and whether it had affected their mindset in respect to PRESTO itself and to development in general. The interviews were semi-structured (see section 3.5.1) and relied on the interview guide available in the appendix (section 10.3.2). I then transcribed and thematically analysed the interviews (see section 3.5.2). In this case, the themes I had in mind before approaching the analysis were the focal points listed in the framework (see section 6.2). During the thematic analysis, these themes were refined and modified, as typical of this method: the presentation of the results in the rest of this section will therefore be structured accordingly.

7.5.2.1 Outcomes of deliberated improvements

First of all, interviews aimed at investigating to what extent the training had supported Delta in answering the questions of the framework, i.e. in shaping their way to integrate UCD and Agile; in addition, interviews enquired on the effect of the training on the organisational structure, work practice, and lifeworld levels (see section 2.3.2) of the company. The following paragraphs will detail different aspects of reported outcomes.

Internal communication. The introduction of UCD and Agile techniques facilitated communication within the team, allowing developers to explicitly articulate and share each one's assumptions on the user and the project goals. Developers acknowledged that communication within the team was now more frequent:

D2: "All of the techniques were cool, mainly because we finally got together to talk"

D4: "[Activities] promoted a participatory, shared discussion' 'I enjoyed being able to know what my colleagues thought […] we share the office with other colleagues as well, so we talk little not to disturb anybody"

Some participants suggested that the training had been particularly effective because of its stepwise structure:

D3: "I think that shaping the workshops as a series of incremental steps has improved the quality of the outcome, both because we had more chances to discuss together, and because we could refine the analysis over time"

D5: "I believe that all of the steps were necessary to arrive where we are […] workshops were fundamental for achieving such a result"

More frequent communication has improved the understanding of project dynamics for all team members and particularly for those who joined most recently:

D5: "It helped me understand a lot of things I had not understood. There's more interaction now"

D3: "I believe that the process has fostered a shared awareness in the team about project goals and what needed to be developed […] it has positively affected our way of working: the project status is much clearer, as is the workload of each of us; we have been discussing on how to organise our work, and we have the chance to promptly intervene on it"

Group awareness and shared understanding of project vision. Quotations also suggest an improvement in group awareness (Gutwin et al., 2004). Richer communication, reinforced collaboration, and shared assumptions over project goals in fact seem to have resulted in a more inclusive team, where fresh members could feel more entitled to ask questions, and challenge pre-existing decisions.

D3: "To me, daily meetings improved team cohesion. I just joined Delta a few months ago, so [daily meetings] provided a welcome chance of discussion with the other developers. Also, I feel less awkward in knocking at their office during the day"

Improved communication and group awareness also resulted in a more grounded and shared understanding of the project and of its underlying vision:

D3: "I had the feeling that we were progressively perceiving PRESTO as a whole product rather than as a collection of non-integrated parts' 'The process did modify my understanding of the project. I finally had the chance to listen to everybody's opinions. This helped me clarify the most relevant aspects […] and the ambiguous points that I had found in the project documentation"

D1: "The real value of this experience is not in the product per se, rather in the process, which led to shared goals, to an understanding of the project direction, and fostered an elaboration process in the team. Now everybody has a shared vision over the project, even though I had already envisioned this same prototype almost two years ago"

Supporting collaboration. Participants sometimes reflected on the lack of a proper design and development methodology in their usual working practice:

D2: "I was a bit sceptical about daily meetings. We used to have periodical meetings in the past, but they were often unnecessarily lengthy and technical, so I was afraid it would end up in the same way"

D3: "We used to have weekly meetings about the general goals of the project […] I would receive some guidelines there, but in a quite unidirectional fashion"

D1 explained this situation as follows:

D1: "Given the small size of the team, we never introduced a formal development methodology […] each developer does his part and then occasionally there is a discussion. We do not have the resources or the time to introduce a more structured approach. Actually, we never had the time to do a proper design either: we start building something and then we see how it looks like – it's rapid prototyping. I reckon however that this, plus the lack of a strong check with the customer, has led to the critical situation we were in"

At the same time, D1 acknowledged that relying on him or on D2 to coordinate the team was not an optimal solution, as both of them were usually busy, had to deal with several projects, and could often be out of the office. This situation, together with the absence of a designer or of a UX-knowledgeable developer, resulted in a lack of design, intended not only in the sense of a usable interface, but even as system architecture:

D1: "We would coordinate through sketches and long conversations […] we never had a designer here, which was one of the issues: each developer would design his own little piece with ample freedom and little overall organisation"

D4: "Before, we would agree on high-level project goals and then each of us would discuss the details of his tasks individually with [D1]. Coordination was not structured… you would just talk face-to-face when and with whom needed"

The proposed activities, which allowed relying on a consolidated design and a clear schedule, seem to have contributed to supporting collaboration within the team:

D5: "[Development meetings] influenced the production process as they precisely defined timing, responsibilities, and deadlines"

D4: "I think [proposed techniques] facilitated coordination"

Impact on the lifeworld level. There is evidence of the training affecting not only the organisational structure level, but also the lifeworld level in Delta. Developers in fact acknowledged a shift in the focus of their mindset from technology to its user:

D2: "Now, when I am working on a graphical interface, I reflect on its quality criteria. Once I would have just added an endless list of buttons and said 'You search them'.' 'In a new project we are working on now, we do not have a customer. In this case, you cannot do as you wish: you have to come up with a persona and ask yourself, what would she do?"

D5: "I was not expecting some elements to be so relevant, e.g. the position of buttons… then I understood it was fundamental"

D3: "As technicians, we would spontaneously design a more complex interface. Reflecting on the final user, we actually decided that a minimal interface was better"

Such effect had already become apparent during workshops, as illustrated by the following utterances:

D3: "Do you think our user cares about all this?"

D2: "Carlo is not capable of doing this!"

(while discussing about a new interface component) *D3: "Do we need this as well?" D4: "I don't know, but we can have it" D3: "But if Carlo does not need it, then it's pointless"*

In general, D1 confirmed an evolution of the working process and most importantly of people participating in it:

D1: "To me it looked like a very sensible approach and it did influence our way of working […] there has been a change in the approach to work, there is more communication within the team, and I can now see a general sensitiveness that not everybody had before […] I think everybody gained something from it. A fresh start for new projects"

7.5.2.2 About proposed techniques

Specific questions asked participants for feedback on the techniques used to translate the framework in practice.

Personas and prototypes. First of all, most participants agreed on the combination of personas and prototypes as the most effective one:

D2: "Prototypes were the UML I never had!' 'I appreciated the combination of personas and prototypes. It allowed us to reflect in a deeper, more effective way on what was really necessary"

D5: "Carlo Furlan [the chosen reference persona]! A celebrity! We've been talking about him all the time!"

D1: "What really allowed a giant leap forward was the introduction of personas, which I guess help in materialising ideas"

Elaborating personas was anyway considered time-consuming and subjective. This may be due to having performed this activity at the very beginning, when the team had not yet achieved a suitable user-oriented mindset:

D2: "Personas cost me a lot of effort […]"

D3: "In my opinion, we devoted too much time to preparing personas, since the interface is now suitable for different kinds of users. However, I reckon that they have been a good exercise to discriminate what was useful and what was not"

Scoping and awareness. Other techniques were considered useful in narrowing the scope of PRESTO, focusing on its essential aspects:

D2: "Interaction metaphors required us to look more closely at the software architecture and at its limitations' 'Stakeholder networks were helpful in delimiting the context we were addressing […] it has been a huge sacrifice to leave out all the rest of the context, but at least we set some boundaries"

In particular, daily meetings and sprint reviews earned strong appreciation due to their nature of "milestone" discussions, which would ensure an aligned, up-to-date awareness of the evolution of the project:

D5: "Daily meetings straightened a lot of things in here. We were more aware of project status" "Sprint reviews were useful to wrap up and also do some self-critique"

D3: "I appreciated that daily meetings allowed us to promptly spot any emerging issues"

The backlog (i.e., the set of tasks stuck to the Scrum board) was preferred to the more traditional list of requirements elaborated previously, to the point that D2 suggested to eliminate the "translation" of requirements into user stories altogether in future iterations of the training. The backlog was in fact considered to be more informative:

D2: "The list of requirements looked promising because it identified the tasks to do […] Then, in practice its expressiveness was limited and the content of requirements was not so explicit"

D3: "Having a list that not only collects the requirements, but really describes in detail what you have to do, is even more useful because sometimes a requirement is not so clear"

Tangibility. Techniques such as prototypes and the Scrum board, characterised by the production of tangible, physical artefacts that could materialise otherwise abstract concepts, seemed to be particularly appreciated:

D4: "Prototypes finally allowed us to have a tangible outcome […] whereas quality criteria were too abstract"

D2: "[Prototypes allow] having a quick piece of paper to show others, and others understand what's in your mind' 'I can tell the developer – look, this is what we drew with the customer, so this is what you have to do, not whatever comes to your mind"

D5: "Prototypes support a participatory discussion […]. When you start seeing on paper what comes out… it's been really useful"

D2: "I appreciated the tangibility of the Scrum board […] it was easier to update than any other digital alternative we had tried previously"

D3: "Since [with the Scrum board] everybody always had everything under control, coming up with new ideas was much easier, and so was improving the software during development […] the team could self-regulate"

Such tangible artefacts, and more in general design artefacts, were recognised as a sort of documentation that could be referenced during development, contributing in grounding articulation work:

D2: "With the prototypes and the use-case diagrams you have a stable reference to the design, I mean not only to the graphical aspects […] the design is described and frozen. To me, this is the most evident outcome of all the activities we did"

D5: "I could not attend the last meetings, but when I saw the final product I realised that it looked exactly how we had designed it. It was really interesting"

7.5.2.3 Potential for improvement

Interviews also uncovered some aspects of the training that could be improved: this feedback will be detailed in the following.

Issues in project management. D2 commented that, although the Scrum board had proven perfectly suitable for coordination within the team, it lacked the granularity he needed in order to fill in timesheets about the project, requiring him to integrate the use of this artefact with Microsoft Project. Therefore, at each daily meeting, he would ask colleagues how many hours they had dedicated to PRESTO the day before (as several projects were going on in parallel within Delta). Because of this, he argued that the board did not provide sufficient awareness of how much work was left to do:

D2: "The board does not show if you are late"

Indeed, approaches like Scrum suggest supplementing the board with a burndown chart (Schwaber, 2000), which plots open tasks against time left. We did not introduce such chart to participants as we did not want to overwhelm them with too many artefacts, given that they were quite novice in their approach to Agile; however, they spontaneously recognised the usefulness of such an instrument.

Furthermore, all participants voiced their struggle in dealing with different projects at the same time, and within this endeavour their even greater difficulty in providing accurate task estimates. The latter issue is well-known in Agile projects (e.g. (Popli and Chauhan, 2012; Usman et al., 2014)):

D3: "Managing sprints was hard. We had issues in estimating tasks and other activities came up also to compensate for delays. We should try in the future to improve our planning […] meeting deadlines affects everything else. If you are late, then you work worse" "We were not able to meet the sprint deadlines. Estimations were incorrect, and most importantly other projects got in the way. However, having sprints was really useful while facing tight deadlines, so I hope that [D1] will reintroduce them in the next projects" "We haven't been that good at estimating task durations"

Indeed, the Scrum master role prescribes that this figure should protect the team from impediments (Schwaber, 2004) and stressful outside influences (Hill, 2007): however, again because the team was quite new to adopting Agile, we preferred to not introduce this figure to the team, nor to take this role along the one of facilitators.

Theoretical background. Contrasting comments concerned the appropriateness of introducing the theory behind proposed techniques in more detail. Some participants, in fact, wished to learn more about it; others were uninterested in this grounding, considering it as a "designerly" way of understanding project planning as opposed to the more pragmatic approach of developers:

D3: "I would have loved to know more about the rationale for some design choices […] I am not sure that this single case has taught me a methodology that I can re-use. To me, knowing more would improve our skills as developers and also improve team awareness"

D2: "We could not yet see the value [of quality criteria]… at the end of the day, we need to implement efficiency and learnability. We are bricklayers, not architects."

External interventions. Contrasting opinions also referred to the role of an external facilitator. On the one hand, the tension in Delta before our intervention was so strong that it mandated being resolved before any further improvements in PRESTO were possible; additionally, an external intervention required participants to be more explicit in articulating their assumptions and their understanding:

D1: "We asked for an external intervention because the differences between my vision and [D2]'s had become irreconcilable […] my aim was to build a shared consensus rather than forcing a project direction in a top-down manner' '[an external intervention] forced the team to reason on who their target user was, and personas were very useful in this respect; there has been an important evolution that allowed to build an overall vision of the project. Furthermore, [the external intervention] also forced to put in place non-obvious collaborations"

D2: "You need everybody to engage and cooperate. This is easier if you have an external intervention"

D5: "[an external intervention] helped us a lot also to sort things out… we felt less lost"

Furthermore, an external intervention made it easier to keep the team focused and to avoid getting lost in details, especially during daily or sprint review meetings:

D2: "The sprint reviews were really good because they were very precise and focused"

D5: "An external facilitator avoids divagations"

On the other hand, the presence of a facilitator seemed to hold back the team from completely appropriating proposed techniques. For instance, developers did not feel entitled to modify the backlog as needed, or to take full responsibility for sustaining collaboration:

D2: "As we were proceeding, the activities on the Scrum board would generate other subtasks about things we had not considered before, but that were needed to have the system work […] we did not add these activities [on the board]. Maybe we should have had a more systematic approach to system design' 'The Scrum board is still there… not updated' 'We now have weekly, rather than daily, meetings"

Lack of user involvement. Finally, some disappointment was caused by the persistent difficulty in accessing users:

D2: "I was a bit disappointed because we did not have the chance of performing user testing with real users"

D4: "I think we still lack the opinion of real users. We have been discussing on the extent of their involvement, but because of the difficulty in accessing them and because of time constraints, we still miss their feedback, really"

7.6 Discussion

This chapter has described the first iteration of CMD (section 3.4.2) in which I trained a team of developers on the adoption of the framework of focal points introduced in Chapter 6. The framework aimed to be a tool for supporting the organisation in assessing the extent of communication breakdowns in its way of integrating UCD and Agile, to the end of informing an intervention on its policies and procedures. In this study, the problem solving interest of the company concerned putting in place a process that could ensure a rapid, flexible implementation and a narrower focus on the user while suiting the contingencies of Delta as a small enterprise. At the same time, my research interest pertained to the appropriateness of training developers on the framework for supporting their company in shaping the integration of UCD and Agile, and to the assessment of any effects of the training at the organisational structure level (see section 2.3.2).

For what concerns Delta's problem solving interest, the re-designed interface was delivered on time and was more usable than the previous one, as shown by the user evaluation (section 7.5.1) and by the evolution of prototypes over time (Figure 25 to Figure 31): it can be concluded that the training proved effective in improving the usability of the PRESTO interface. It is worth remembering that these mockups were entirely produced by developers, without significant contribution from facilitators. In addition, the training was structured to require limited resources in terms of time and effort, to be at least partially on-site, and in general to be tailored to the needs and contingencies of the company. With respect to the framework (Table 4), the training allowed to identify the user, rather than the product owner, as more fit to the needs of Delta: the company in fact had a well-defined customer, but was struggling in the definition and thus the recruitment of users. Furthermore, the training allowed to make articulation work explicit and documented through design artefacts, and to improve team collaboration.

In terms of my research interest, the effects of the training will be described with respect to the categories of the organisational context described in section 2.3.2 and derived from (Bjørn and

Ngwenyama, 2009). First of all, the training produced an effect at the organisational structure level of Delta. Referring to (Rönkkö et al., 2005), the training could be seen as the introduction of a plan that helps to coordinate development work, identifies parts in software development practice needing support for articulation and re-planning, and makes the gap between the plan itself and ongoing practice visible. The introduction of such a structure over the development process was allowed both by the nature of proposed techniques, drawn from approaches such as UCD and Agile known to be supportive of cooperative work (section 2.2), and by the presence of an external facilitator. An improved support of articulation work among the team also favoured group awareness (Dourish and Bellotti, 1992), making mutual assumptions explicit and favouring their alignment. In this case, the introduction of a tool to facilitate coordination and communication, as suggested for instance in (Guzzi et al., 2015; Halverson et al., 2006), might not have been as effective: this resonates with the area of research on software processes as enactments, which sees the outcome of a software process not only as code, but also as work in terms of culture and social interaction (Cohn et al., 2009).

Furthermore, the training also affected the lifeworld level of the company: quotations from interviews (e.g. D1's reference to a "designer's sensitivity" now pervading the development team) highlight a shift from a "developers' mindset" (Ardito et al., 2014), purely focused on technology, to a more user-centred one. It should be noted that not all of the questions in the framework were answered during the training: the translation of the framework back into practice had to account for the situatedness of the software development practice, since "coherence [of software development] is achieved not by blindly following methodologies but by adapting them to the circumstances of the project" (Procter et al., 2011), and for the specific organisational and social context (Dittrich et al., 2008; Ferreira et al., 2011) of Delta.

Finally, here follow some considerations on the feedback for improvement reported in section 7.5.2.3. First, the training did not mean to introduce UCD and Agile in full, but rather to provide a basic understanding of their principles and techniques, making the company aware of their benefits and providing elements to decide e.g. whether to proceed in developers' training or to hire a specialist designer: this should be considered when addressing issues in project management and contrasting requests on the amount of theoretical background to be presented. Second, the limited appropriation of proposed techniques may be due to Delta not being well accustomed with the Agile mindset, and thus not being used to self-organising their team. Third, the unresolved issue about the lack of user involvement, due to difficulties in accessing users, points to the importance of customer sponsorship as recommended in section 6.3.

8 Study 4: eMaze

This chapter will describe the second iteration of CMD. Also in this study, developers were trained on the adoption of the framework of focal points introduced in Chapter 6; however, the organisation where the training took place was more accustomed to Agile. The training was therefore refined based on this kind of setting and on the outcomes of the previous study in Delta, presented in Chapter 7.

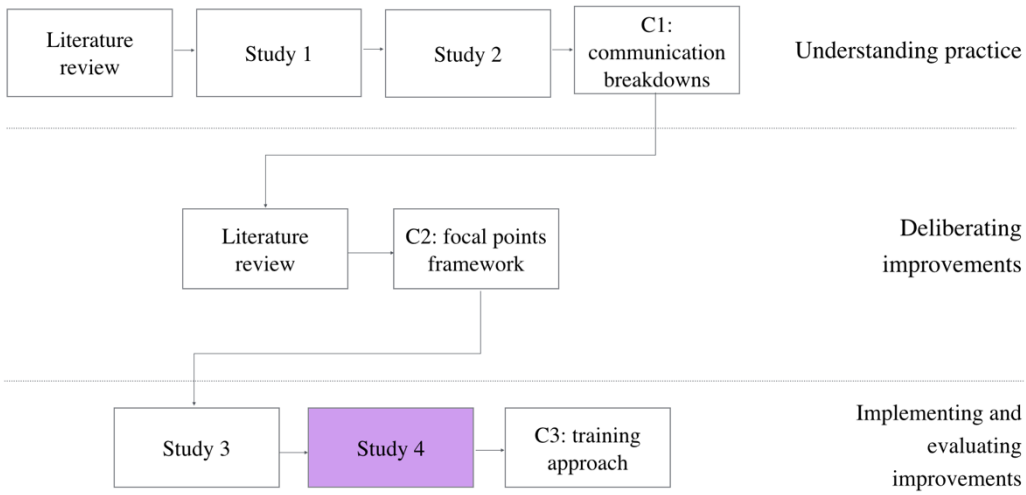


Figure 33 - dissertation progress: study 4

8.1 The context

The study described in this chapter was carried out in one of the five branches of a quite large Italian IT group, called eMaze, which counts about a hundred employees overall and provides cyber security and network configuration services to the largest telecommunication operators of the country. A representative of eMaze approached me explaining that the company had successfully implemented an Agile process since a few years, and wished to transition from a model focused on projects carried out essentially for one main customer (a large telecommunication provider that will be referred to as "the Telco") towards a business grounded on more standardised products, in order to reduce dependency from the Telco. While in the process, however, eMaze realised that they were lacking sufficient skills in usability and interface design, and that this could be an issue in proposing their products to fresh customers. In the terminology of CMD, this expressed the problem solving interest of the company (see section 3.4.2).

Activities were structured as a second iteration of CMD: following the structure of this approach illustrated in section 3.4.2, and similarly to the study described in Chapter 7, they consisted of a stage of understanding practice followed by a stage of deliberating and implementing improvements. The first stage will be described in section 8.2, while the second stage will be described in section 8.3.

8.2 Understanding practice

The understanding practice stage included an interview study with the personnel and the selection of a specific project to be used as a running example. I carried out the interviews at the eMaze site in June 2016: this activity lasted two full days and involved 7 people among developers and managers. As usual, the interviews were semi-structured (see section 3.5.1): in this case, they aimed at gaining some initial insights about the three level of the organisational context in eMaze, namely in terms of its work practice (which methods were applied in day-to-day development), organisational structure (how was the Agile approach intended), and lifeworld (what values drive the company, and how do its relations with the staff, headquarters, and customers unfold). The interview guide is available in the appendix (see section 10.4.1). I then transcribed and thematically analysed the interviews (see section 3.5.2). The main findings are described in the following.

The organisation. eMaze (intended here and in the following as the branch where the intervention took place) employs about 20 people, mostly young graduate developers, and features a pretty flat hierarchy: a few key figures emerge because of their seniority and acknowledged experience. Developers have been using Scrum for years; recently, however, this process has become looser, mainly due to the loss of several leading members as explained later. For instance, pair programming and retrospectives have been abandoned, and meetings have become less frequent and periodical. The environment is predominantly technical and does not include designers or otherwise UX-skilled personnel, yet employees showed a positive and rather curious attitude towards UCD-related themes, to the point of explicitly arguing in favour of the training in front of the managers of the whole group.

Relationship with the headquarters. The headquarters of the group are located in another city quite far from the branch at hand. Recently, the ownership of the whole group changed, and the new owner imposed a micromanagement approach over branches, introducing techniques to monitor the productivity of teams. This disrupted the relationship between the headquarters and the branch, where employees were accustomed to a flexible, autonomous management of their own job inspired by the Agile values. In fact, developers interpreted this new structured

approach as a lack of trust in employees and in their accountability, to the point that, as mentioned above, several senior people left the company within a few months.

Relationship with the customer. The Telco is eMaze's main customer: being a much larger venture, the power relationship between the two parties is quite asymmetrical, although it seems to have reached a stable and reasonable balance over time. Yet, several employees at eMaze reported difficulties in interacting with representatives of Telco, a rigidly structured corporate where communication between departments appears to be limited and whose highly constrained workflow prevents a full implementation of Agile in the projects followed by eMaze (e.g. for what concerns continuous delivery). In particular, a few developers highlighted that representatives of Telco, usually belonging to their IT or marketing departments, often demonstrated resistance in allowing contact between eMaze and final users of their software.

In the spirit of deliberating improvements with practitioners, and of applying the training to a software that would be familiar and significant to them, I asked eMaze to identify a specific project that could be used as a running example. A representative suggested to focus on what will be referred to as "the Software", a desktop application used to configure and monitor networking devices for corporate customers of Telco. The Software is used by a variety of employees of Telco and affiliated companies to manage different stages of a complex workflow, and was chosen for the following reasons: it is entirely developed within eMaze for Telco, which however does not influence its design (in several projects the Telco imposed an user experience already elaborated by its marketing department); it has evolved over the years as the juxtaposition of different parts, and would now need a major refactoring; being one of the main projects of the company, it is sufficiently well known to all employees.

8.3 Implementing improvements

Based on the insights gained from the understanding practice stage of this study, I designed the structure of the training as composed of the four workshops illustrated in Figure 34. Activities emphasised the UCD perspective over the Agile one, due to the familiarity of the company with Agile and to its strong interest in understanding how to deal with a user, what feedback can be obtained, and how to elicit it. With respect to the framework of focal points (Table 4), the role of the product owner of the Software was already clearly played by representatives of Telco, hence the workshops focused on the identity of the user instead. Addressing this interest of the company would be useful both for justifying with the Telco the need for accessing any users, and for improving the quality of products when targeting fresh customers. In addition, the focal point of articulation work was interpreted as concerning how different sorts of design artefacts, from personas to prototypes, could support communication within the team and with the

customer. As what done in Delta, and following the guidelines of CMD (see section 3.4.2), contents adjusted over time to accommodate participants' interests and outcomes of each workshop. As already pointed out in section 7.6, also in this case not all the questions of the framework were answered through the training, as not all of them seemed appropriate: this is because of the specific characteristics of the company and more in general of the need to account for the situatedness of the software development practice.

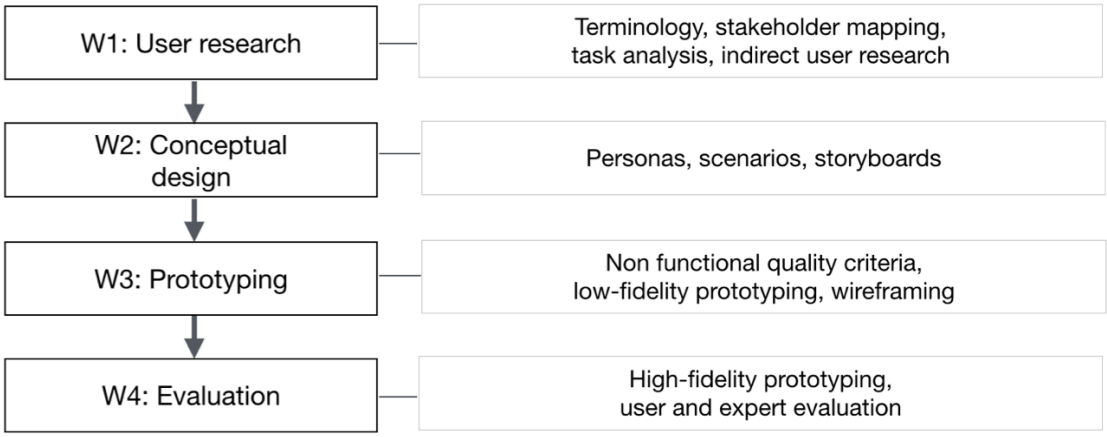


Figure 34 - overview of workshops in eMaze

The detailed schedule of the workshops, all run at the eMaze site in August 2016, is outlined in Table 8.

Table 8 – schedule of workshops in eMaze

Workshop	Date	Duration
W1: user research	02/08/16	7 hours
W2: conceptual design	19/08/16	7.5 hours
W3: prototyping	24/08/16	7 hours
W4: evaluation	25/08/16	6.5 hours

Three developers were appointed to attend the whole set of workshops; occasionally, a couple of other more senior colleagues joined. In order to respect their confidentiality, participants will be referred to as D1 to D3. All of them expressed great interest in user-related themes: D1 was moderately self-taught on UCD techniques, while D2 and D3 were not familiar at all with them.

D3: "One doesn't even know where to start from, without knowing any basics"

D1: "More than once [design choices] have been a stab in the dark"

D2: "If there's one skill in the Company we are really lacking, it is interface design, because on many projects we have no freedom on that and we could not gain experience… we try to do what we can"

8.3.1 Workshop 1 – user research

Although participants had demonstrated a positive attitude towards UCD themes, activities started by motivating more formally the benefits of this approach, for instance by discussing well-known reports from the industry (Kumana and Dickinson, 2014; *The Standish Group's CHAOS Report*, 2014) that highlight user involvement as one of the key factors for project success, and in contrast the lack of it as one of the most common reasons for failure. Basic concepts of UCD (e.g. stages of the design cycle, definition of usability) were also explicitly introduced to ensure an aligned understanding of the core tenets of the approach among participants.

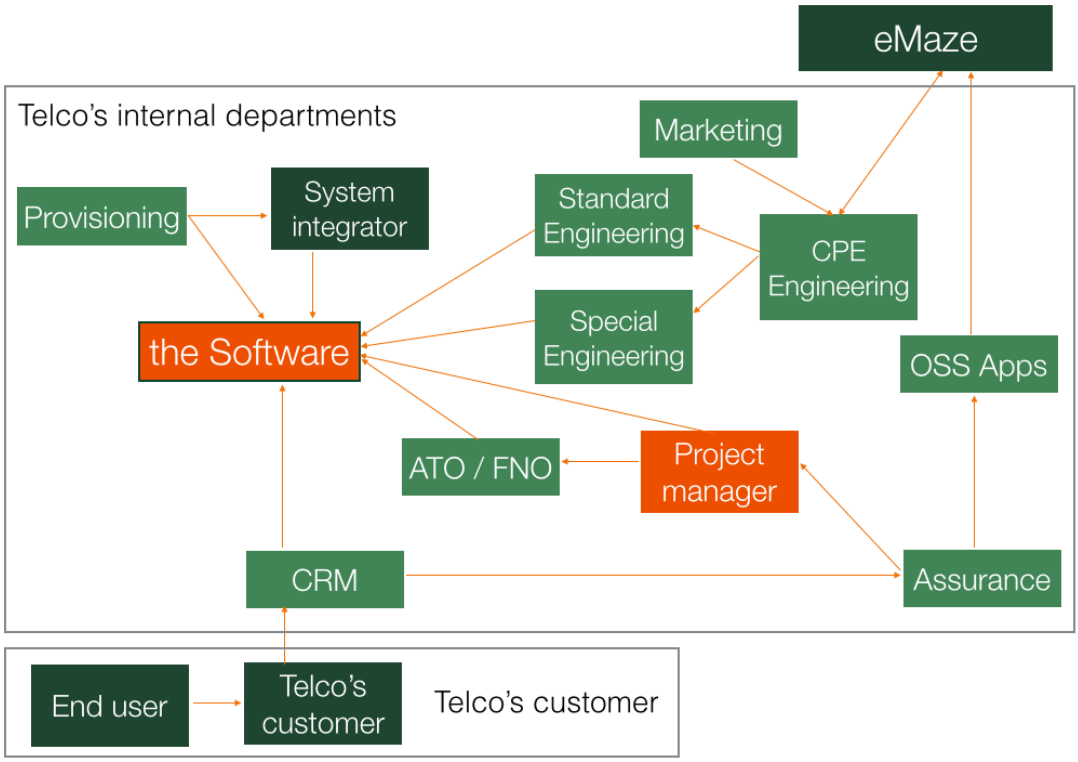


Figure 35 - communication network in eMaze

We then turned to analysing the workflow supported by the Software, graphically represented in its user manual. It was apparent that, despite being extremely well specified, such workflow was heavily technology-centred rather than user-centred or even business-centred: in other words, it reflected the internal functioning of the Software, rather than the tasks to be

accomplished by the different kinds of users meant to interact with it. Therefore, participants were guided in re-elaborating the existing representation of the workflow, focusing on the perspective of the people interacting with the Software at different stages of the business process. In this way, relevant actors began to be outlined more precisely and participants clarified the terminology specific to the Software. Once the workflow was re-elaborated, I asked participants to detail the relationship between the different actors identified and eMaze: to visualise these information in a structured way, we collaboratively drew the communication network (Cataldo and Herbsleb, 2008) shown in Figure 35.

The communication network shows that lots of actors use the Software, yet they are not the same people who directly speak with eMaze; because of this, their needs, if collected, are inevitably mediated when reported to the company. At this point, we focused on the actors who looked more likely to truly interact with the Software, namely those directly connected to it in the communication network, and participants performed a task analysis of what these actors were meant to do using the Software. Such information was formalised through use-case diagrams drawn on a whiteboard; participants then agreed to concentrate on the project manager (PM) for the rest of the activities, as this type of user seemed not only likely to interact substantially with the Software in daily work, but also to be using a variety of features with respect to other actors. The use-case diagram related to the PM is illustrated in Figure 36.

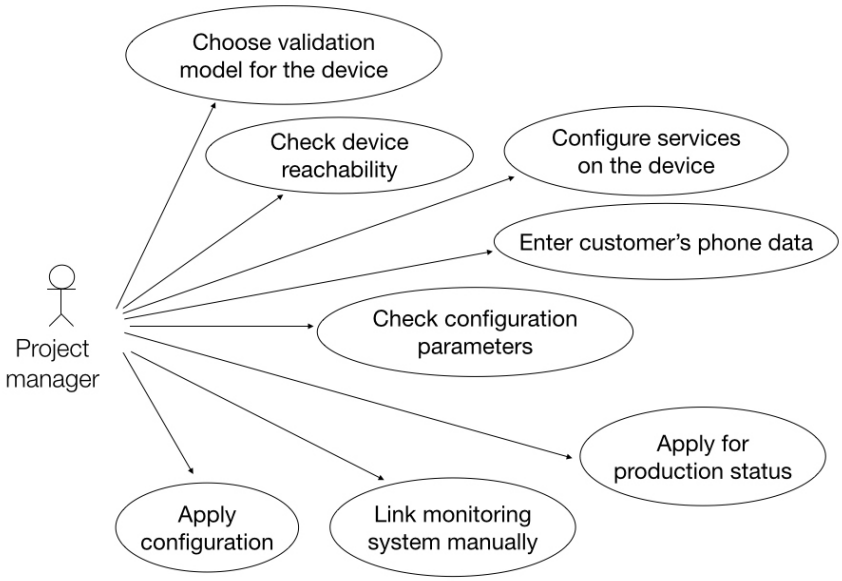


Figure 36 - Use case diagram for the project manager (PM)

Talking to real users was almost unattainable for eMaze because of the opposition of Telco intermediaries: hence, developers needed to find alternative ways of collecting information on which to ground their understanding of users and in turn to guide software design; in other

words, they had to come up with "indirect" techniques of user research. Once the PM was chosen as the reference user, participants suggested inviting one of their managers to join the discussion, since this person often interacted with customer representatives and was therefore more aware of the internal dynamics of Telco. The manager confirmed difficulties in accessing users and advocated a stronger narrative to persuade Telco representatives that allowing eMaze to meet users would not threaten their status, but rather improve the quality of the service offered. Furthermore, he added more details on some aspects of the communication network, such as characteristics of mentioned users, the high turnover in several departments, and the subsequent lack of training of operators that interact with the Software. Participants agreed to attempt an indirect user research and to report their results during the next workshop.

8.3.2 Workshop 2 - conceptual design

At the beginning of the second workshop, D2 and D3 reported the outcomes of their indirect user research activities, explaining that they had retrieved the list of user accounts registered on the Software, had been able to deduct the names of a few project managers from them, and had checked related profiles on LinkedIn, extracting information on the background of these people (usually technical) and on their experience in working for the Telco (in several cases remarkably long). These information, supplemented with the manager's comments from the previous workshop, were then used to build the following two personas (Cooper, 2004) representing different profiles of PM:

- Gianluca, a junior PM: he has little experience and motivation, and interacts with the Software to perform lots of simple activities for a number of clients;
- Lucrezia, a senior PM: she is strongly motivated, committed, and skilled, and interacts with the Software to perform several high-responsibility tasks for the few high-stake clients she follows.

In reference to the framework, these personas were meant to document the outcomes of user research, supporting articulation work and collaboration at the same time by allowing findings to be shared with the rest of the team.

For both Lucrezia and Gianluca, participants elaborated a few scenarios related to the tasks identified in Figure 36; these scenarios were then translated into storyboards, either composed digitally using the tool StoryboardThat (http://www.storyboardthat.com/) (Figure 37) or hand-drawn (Figure 38).

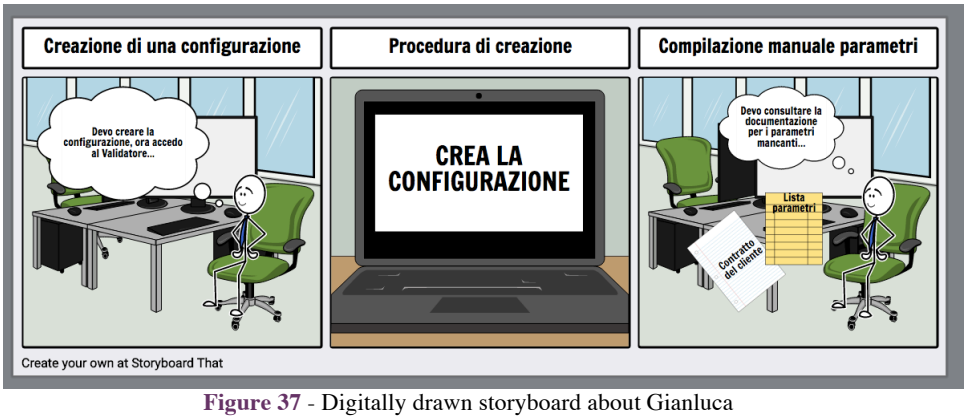


Figure 37 - Digitally drawn storyboard about Gianluca

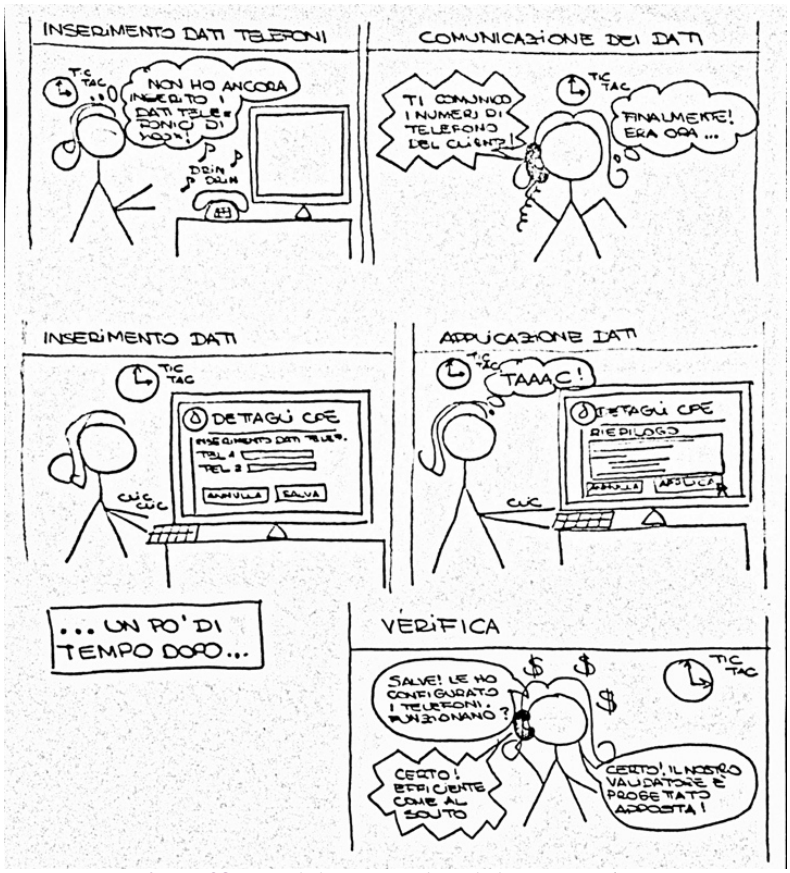


Figure 38 - Hand-drawn storyboard about Lucrezia

8.3.3 Workshop 3 - prototyping

In order to simplify subsequent activities, participants chose to concentrate on Lucrezia as the reference persona representing a PM using the Software. Then, the workshop started by discussing non-functional quality criteria for the Software. The five dimensions of usability listed by Nielsen (Nielsen, 1994) were proposed as an example of these; participants were asked to rate how relevant each of them would be to Lucrezia by applying the planning poker (Grenning, 2002) technique borrowed from the Extreme Programming approach (Beck, 2000)

(Table 9). In reference to the framework, all of these activities still belonged to a hypothetical Sprint 0, and were always closed with a tangible outcome that could be later shared with the rest of the team to facilitate collaboration.

Table 9 - quality criteria for the Software

Dimension	Definition	Priority
Learnability	How easy is it for users to accomplish basic tasks the first time they encounter the design?	2
Efficiency	Once users have learned the design, how quickly can they perform tasks?	**5**
Memorability	When users return to the design after a period of not using it, how easily can they reestablish proficiency?	**4**
Errors	How many errors do users make, how severe are these errors, and how easily can they recover from the errors?	**5**
Satisfaction	How pleasant is it to use the design?	3

The concept of interaction metaphor was introduced next as a mediator between the system and the user: in particular, we considered the re-design of a specific form in the Software that allowed the PM to choose the appropriate device configuration depending on what specified in the customer contract. Three alternatives were explored and rapidly prototyped through sketches, as exemplified in Figure 39: referring to the framework, this set of low-fidelity prototypes was meant to document design choices and their rationale, thus supporting articulation work.

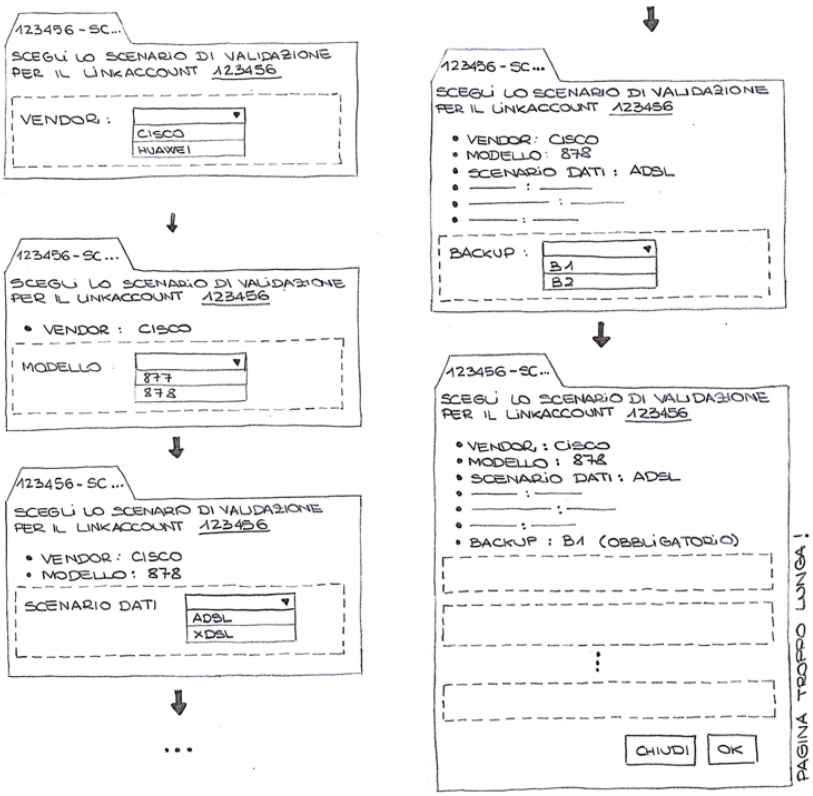


Figure 39 - Low-fidelity prototype

The prototypes were then compared to the existing interface, for instance by performing cognitive walkthroughs, and participants collaboratively reflected on whether any prototype would improve the interaction of the PM with the Software in respect of the quality criteria agreed in Table 9. It turned out that the complexity of the current interface was anyway capable to support the elaborate workflow related to device configuration better than proposed alternatives; therefore, and also due to time constraints, we chose not to iterate over prototypes, but rather to reconsider the existing interface, which was functionally effective yet little usable. To this end, we focused on the main form for managing each specific network device: such form had in fact evolved substantially over time with the addition of a lot of features not always coordinated. Participants were guided in a high-level analysis of the business logic behind the form, identifying logically similar functionalities and re-grouping them in a wireframe drawn on the whiteboard. This adaptation of the training activities resonates with the training not being meant to introduce the full UCD or Agile methodologies, but rather providing guidance on the adoption of the framework, and with the recommendations of CMD about involving practitioners in shaping the intervention.

8.3.4 Workshop 4 - evaluation

The last workshop introduced high-fidelity prototyping; in particular, since D1 repeatedly pointed out that Telco would not accept discussing over a "non serious" low-fidelity prototype and that previous attempts at doing this had failed, the purposes of low- and high-fidelity prototyping were detailed and the need for a correct communication of such purposes to the customer, especially if she was not familiar with UCD approaches, was emphasised, partly addressing the section of the framework related to articulation work. Finally, elements of user evaluation were introduced. Participants wished to see some of these techniques in action, but were not expecting to be able to apply them personally with users: therefore, they proposed to simulate the user testing of the ERP system used in the Company, which they regarded as particularly cumbersome. Being the only one unfamiliar with the system, I played the role of the user, while D3 acted as the researcher instructing me on the tasks to perform with the system.

Finally, participants invited one of their managers for a wrap-up session, in which they summarised the activities carried out during the workshops and the rationale behind them, and showcased elaborated artefacts. In addition, a month after the end of the workshop cycle, participants shared with me the slides they prepared in occasion of a dissemination seminar they led in front of the rest of the Company.

8.4 Evaluating improvements

In December 2016, workshop participants were interviewed to evaluate implemented improvements. The problem solving interest of eMaze was to acquire the UX skills needed to develop products appealing also to fresh customers: however, a few months after the intervention the company had not had the chance to start any project of this kind. Therefore, it was not possible to assess whether the problem solving interest of the company had been fully answered. Instead, I aimed at assessing whether the training had produced any impact at the work practice level of the company instead. With respect to the study in Delta (Chapter 7), my research interest remained the same, and concerned the appropriateness of training developers on the framework for supporting the shaping of the integration of UCD and Agile. In addition, and based on the outcomes of the study in Delta, I also aimed to assess whether the training had produced any effects not only at the organisational structure level of the company, but also at its lifeworld level.

The semi-structured interviews (see section 3.5.1) were carried out by external researchers to minimise social bias and were based on an interview guide available in the appendix (see

section 10.4.2). I then transcribed and thematically analysed (see section 3.5.2) the interviews based on the themes emerged from the study in Delta. Findings will be discussed in the following. All quotes have been translated from Italian.

Participants welcomed the training as a chance of professional growth (typically referring to it as "the course"):

D2: "In terms of common sense, this is what every developer should do. Yet having someone explaining you the steps to follow is something different"

D3: "Now I have a method"

Both participants and the management expressed the unanimous intent to apply proposed techniques in upcoming projects where a fresh interface will be required:

D2: "I have been assigned to a project where the interface is set in stone [by Telco], BUT we all agree that we are going to apply UCD techniques at the first suitable occasion"

D1: *"If we had a different customer, I'd have already applied all the techniques we talked about"*

Even a few months after the training took place, participants generally remembered correctly proposed techniques and their rationale, and acknowledged a shift in their focus from the technology to the user:

D3: "Before we used to say – [the user] will have to get over it" "The interface as the means to achieve an objective from the user's point of view" "The goal is to remove the need for a manual – even for us as developers!"

D2: *"We'll surely follow this approach rather than – bah, let's just do something"*

Even though, since the end of the training, no project requiring a major interface design had come up, D2 and D3 claimed to have applied proposed techniques to the improvement of parts of the Software interface that had been assigned to them. Furthermore, participants repeatedly mentioned how learnt techniques were now in use in their daily work practice:

D1: "I enjoyed wireframing a lot. It really gave me a different point of view" "By agreeing on quality criteria you can make subjective aspects more concrete and factual"

D3: "We should apply the wireframing to a re-design of several existing components…we need to organise the information or the poor user will be scared"

In particular, they suggested that presenting generated artefacts to their colleagues contributed to supporting internal communication:

D3: "Prototypes and scenarios can be used internally to understand how to design something […] I have proposed some prototypes to my colleagues and I believe this simplified the design and the discussion" "In my opinion personas should be shared by the whole team… they can be used to raise awareness among colleagues"

D2: "I tried to retrace some of the steps in the activities where I could"

Participants also commented on the positive attitude shown by the rest of the Company at the end of the dissemination seminar they led; D2 even claimed having described the activities followed to friends working in other ventures.

D2: "We reported to the rest of the Company and the reaction was – let's hope we will soon have projects where to apply this approach"

Despite the satisfaction and the interest shown, however, participants underlined how in their opinion the effects of the training would be limited by the strong "developer mindset" (Ardito et al., 2014) of Telco's representatives, even harder to overcome due to the unbalanced power relationship with eMaze:

D1: "Personas cannot be used with Telco: our customer is very much feature-oriented and in a dominant position […] Our customer does not want to see the prototype; it wants to see the product" "Storyboards look too playful to serve in our communication with the customer" "Some techniques will be more applicable than others, because it is impossible to access users […] We have no [user] feedback. Clearly we miss it"

D3: "I guess the customer would be disappointed by storyboards on paper. We make software, and he may think we did not put too much effort into such a proposal" "Evaluation techniques will be more useful to us as a retrospective, or as a self-evaluation… we have never met our users"

8.5 Comparison with Delta

This chapter has presented the second iteration of CMD (see section 3.4.2) in which I trained a subset of the eMaze developers' team on the adoption of the framework of focal points to steer the integration of UCD and Agile. The problem solving interest of the company was to acquire the UX skills needed to develop products appealing also to fresh customers; my research interest instead concerned, as in the previous iteration (Chapter 7), determining whether the

training effectively supported the company in integrating UCD and Agile and had any impact at the different levels of the organisational context (Bjørn and Ngwenyama, 2009).

With respect to the previous study carried out in Delta (Chapter 7), this intervention occurred in a more mature setting: eMaze was in fact long accustomed to Agile and to a structured development process, and exhibited a genuine, widespread interest in enriching it with UCD (or, in other words, a "culture receptive to UCD", as explained in Chapter 5.3). Because of these contingencies, the training needed to be tailored differently from what done in Delta to remain meaningful, accounting for the situatedness of software development practice (Rönkkö et al., 2005): therefore, the training in eMaze mainly emphasised UCD.

Another difference with Delta was that participants in eMaze only represented a subset of the development team, rather than its entirety: yet, these participants acted as ambassadors, disseminating what learnt to the rest of the organisation, for instance by introducing colleagues to learnt UCD techniques, or presenting training results to the rest of the team in the final wrap-up session. This can be seen as an effect at the organisational structure level (Bjørn and Ngwenyama, 2009). With respect to the framework of focal points, I argue that involving a subset of a cohesive development team in the training has proven sufficient to reinforce the areas of cooperation and articulation work.

In eMaze, the widespread effect of the training exceeded the impact observed in the study carried out in Delta: not only did the training affect the organisational structure level and the lifeworld level, but also the work practice one. As mentioned above, the training had an impact at the organisational structure level in terms of enhancing articulation work and cooperation. In addition, and as witnessed by evaluation interviews (section 8.4), it also caused changes at the lifeworld level, shifting developers' perspective from being technology-centred to being more user-centred. Moreover, the same interviews in eMaze suggested a deeper appropriation of proposed techniques also at the work practice level: see for instance the reported use of prototypes to support cooperation in the evaluation interviews. Artefacts such as prototypes and personas, which are produced applying techniques proposed during the training, became boundary objects similarly to the briefs discussed in section 5.3: in fact, they became a common reference for different people working at the same project, supporting cooperation and communication.

8.6 Training on the framework adoption

The two studies described in Chapters 7 and 8 instantiated a training approach on the adoption of the framework of focal points that represented contribution C2 of this thesis. The training served to implement and evaluate the improvements proposed by means of the framework, which identifies focal points to diagnose and assess the extent of communication breakdowns within the organisation. The two studies allowed to implement and evaluate the training, which can now be regarded as the third contribution of this thesis (C3, as shown in Figure 40) and as the answer to research question RQ3 ("how can the solutions identified in RQ2 be adapted to the specific setting of small organisations?", see section 1.2).

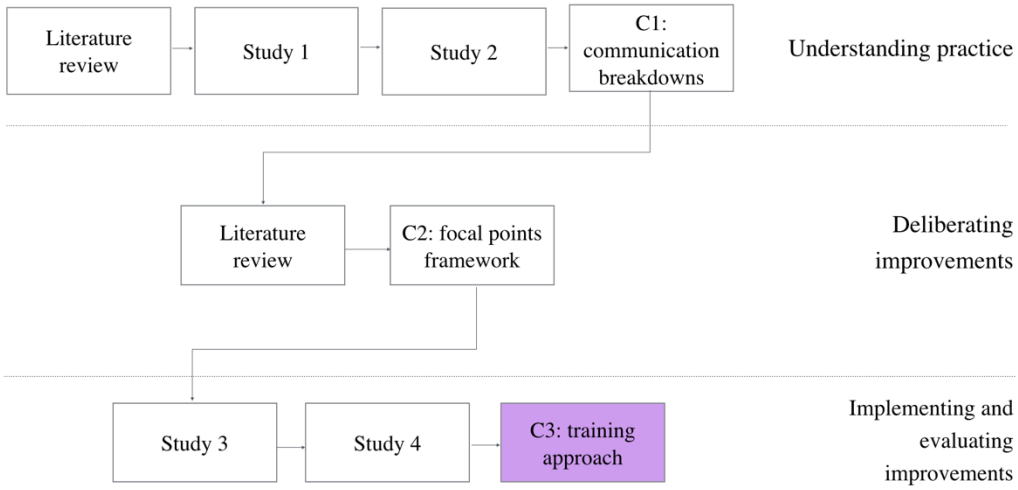


Figure 40 - dissertation progress: contribution C3

The effects of the training are summarised in Figure 41. With respect to the categories of the organisational context introduced in section 2.3.1, this intervention is positioned at the organisational structure level, for the reasons detailed in section 6.3; however, the training affected all of the three levels. Specifically, it reinforced cooperation and articulation work in terms of the organisational structure level; it fostered the adoption of UCD techniques in terms of the work practice level; and it encouraged a shift towards a user-centred mindset in terms of the lifeworld level.

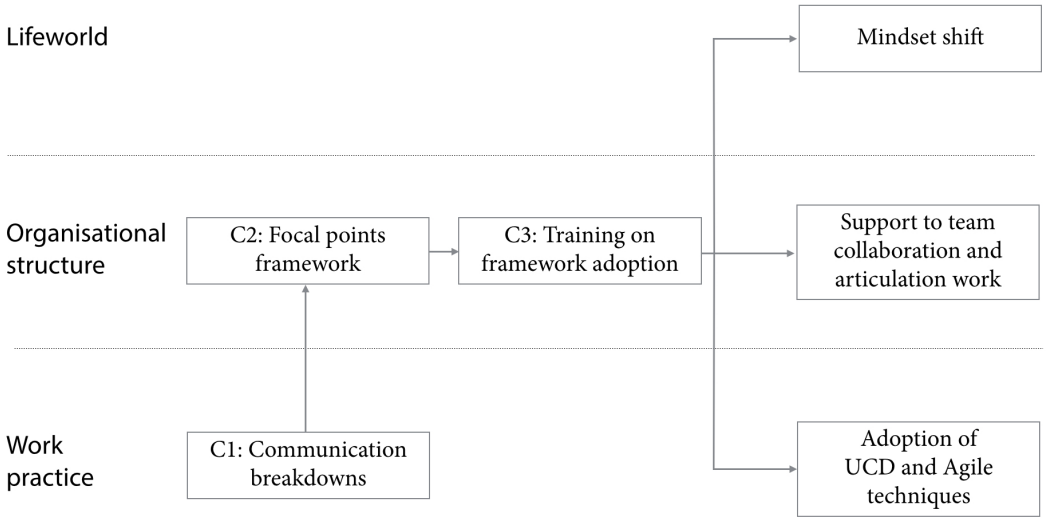


Figure 41 - effects of the training

In eMaze, it is likely that the familiarity with the Agile mindset, in which developers are seen as capable of self-organisation and micro-management is discouraged, caused developers to readily adopt the UCD techniques they perceived more valuable for instantiating their mindset shift in daily practice: in turn, this appropriation is likely to favour the sustainability of the effects of the training, i.e. of organisational change, in the long run. A more mature organisational culture may also explain why the presence of a facilitator was seen in eMaze as a resource to support learning rather than as a factor limiting ownership, as it partially happened in Delta. In both cases, however, an external facilitator was perceived as useful in enforcing focus on the user and constraining digressions.

Nonetheless, the impact of proposed techniques seems to be limited by the fact that eMaze does Agile in a non-Agile environment, where this label includes both the culture of the owner group and of the Telco. Issues of scepticism in customer involvement or of fear of loss of control are rather common in such scenarios, with literature reports on the strategies used to change the management's mindset and the organisation culture (Gregory et al., 2016; Hoda et al., 2011; Kuusinen et al., 2016). I argue that the same challenges encountered in doing Agile in a non-Agile environment can be found when introducing UCD in an environment that is not accustomed to it and is actually resistant. As already recommended in section 7.6, this stresses to the research and practitioners' community that there is still a lack of suitable ways to clearly communicate to reluctant customers the potential benefits of Agile and UCD (Hoda et al., 2011), such as a set of actionable measures that can more objectively assess the positive impact of user involvement on the quality of produced software (Mao et al., 2005).

9 Discussion

This thesis has been about understanding the integration of UCD and Agile and supporting its adoption in small enterprises. To this end, it has followed a qualitative approach inspired by the CSCW and the software engineering communities. This approach is composed of three stages: understanding practice, deliberating improvements, and implementing and evaluating improvements (Figure 42). This piece of work contributes to the body of empirical research on the integration of UCD and Agile, whose extent is still limited (Brhel et al., 2015; Dybå and Dingsøyr, 2008); more broadly, it also contributes to qualitative research on software development. Results consisted of three contributions: a set of communication breakdowns that may affect the integration of UCD and Agile (C1); the translation of this set into a framework of focal points to support organisations in adopting the integration of the two approaches (C2); and training to facilitate the adoption of the framework (C3). This chapter will retrace the activities carried out within each stage, particularly focusing on the research questions that motivated this thesis and the contributions that answered them.

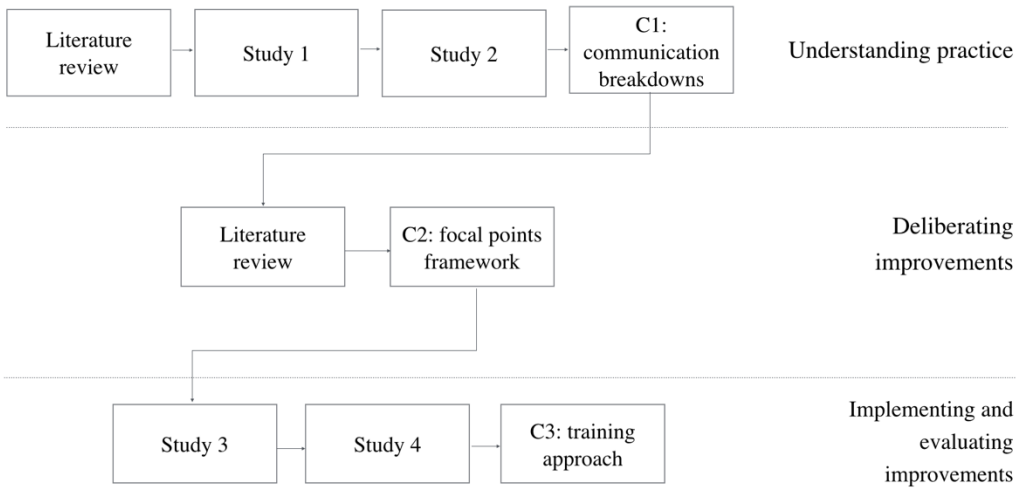


Figure 42 - dissertation outline

The understanding practice stage answered research question RQ1, which concerned the identification of the aspects of work practice where difficulties in the integration of UCD and Agile appear. This was done by performing a literature review that allowed to pinpoint a set of sensitising concepts representing critical issues in the integration of the two approaches (Chapter 2). These sensitising concepts then provided a general orientation for the analysis of the data collected in Study 1 and Study 2 (Chapters 4 and 5). At the end of the two studies,

sensitising concepts had been progressively refined into a set of communication breakdowns in the integration of UCD and Agile, namely: user vs customer, role of documentation, and synchronisation of design and development. This set of communication breakdowns constitutes the first contribution of this thesis (C1) and synthesises the findings of empirical research on the practice of the integration that I either carried out personally or that was reported in literature, as extensively described in section 2.3.2.

The process that led to the consolidation of communication breakdowns in contribution C1 strongly convinced me that the solution to critical issues in the integration of UCD and Agile was not likely to be found in the introduction of a tool, as I had initially envisioned while planning my PhD and as suggested by several authors in the CSCW community (see section 2.2). Rather, the solution was more likely to effectively take place if sought in an improvement of the organisational environment, as advocated in section 6.1.1. I believe that this is the main lesson that I learned during this research work: both UCD and Agile, and even more so their integration, express their potential at the fullest when enacted by people who truly share their underlying values and support them beyond the mere adoption of practices. Tools can only be useful to a limited extent if the team and the whole organisation do not exhibit a rich and pervasive communication, both informal and structured, ensuring that information is shared and coordination is effective.

The deliberating improvements stage aimed at answering research question RQ2, which concerned the identification of practical solutions for the communication breakdowns listed in C1. It resulted in contribution C2 (Chapter 6), which refers to the framework of focal points to drive the integration of UCD and Agile. The framework represents an abstraction of the communication breakdowns from the work practice level, where they become apparent, to the organisational structure level of the company, where their root cause may lie. The focal points are the following: product owner, articulation work, team collaboration. Each of them is articulated in a list of questions to help the organisation diagnose and assess the extent of communication breakdowns within its work practice.

Practitioners may well peruse the framework in its current form, at least as a diagnostic tool to assess the status of the company with respect to the integration of UCD and Agile. However, this thesis draws heavily from the practice paradigm (Kuutti and Bannon, 2014), hence I also aimed at elaborating a more actionable contribution on top of the framework in C2. Research question RQ3, in fact, concerned how the solutions identified in RQ2 could be adapted to the specific setting of small businesses, hence accounting for the peculiarities of this organisational context such as limited resources in terms of funding, available skills, and personnel. The framework was therefore further elaborated into a training approach on the use of the

framework itself. The training was meant as a non-directive intervention, where change was sought indirectly (Baskerville and Wood-Harper, 1996, p. 238); in addition, it had the goal of facilitating the adoption of the framework and the answers to its questions, supporting each organisation in devising its own way of integrating UCD and Agile while being mindful of the characteristics of its specific organisational context.

The implementing and evaluating improvements stage instantiated the training in two iterations of action research, i.e. in Study 3 and Study 4 (Chapters 7 and 8 respectively): after these two studies, the training was sufficiently consolidated as to constitute contribution C3. The two studies proved successful in addressing the research interest of assessing the appropriateness and effectiveness of training developers on the use of the framework to drive the integration of UCD and Agile in their organisation. The outcomes of the two studies contribute to the still limited body of empirical knowledge on training developers on basic UX tasks (Øvad et al., 2015; Øvad and Larsen, 2016b) and confirm several findings reported in literature. First of all, shaping the training as a brief series of hands-on workshops proved applicable even in a different organisational context than that described in (Øvad and Larsen, 2016b, 2016a), which referred to large, global ventures already including a dedicated UX department or a considerable R&D team. Moreover, developers appreciated such an approach, especially because of its pragmatic orientation, confirming the findings in (Øvad et al., 2015). Second, the studies confirmed that "real life tasks were advantageous to apply and quickly showed the developers how and where the method could be used in their own work" (Øvad and Larsen, 2016a, p. 1088). Third, as reported in (Øvad and Larsen, 2016b), the training not only enhanced developers' skills, but also consolidated their supportive attitude towards UCD providing more factual elements for grounding it. In agreement with Øvad's work, these two aspects enabled a variety of positive changes in the organisational environment, such as the establishment of a shared language in the team (Øvad and Larsen, 2016b), an increasing awareness and understanding of usability (Øvad et al., 2015; Øvad and Larsen, 2016a), stronger confidence and self-efficacy (Øvad et al., 2015), and a clearer holistic view of the use of the software and of the core business of the company (Eriksson et al., 2009).

These effects can be positioned at the organisational structure and lifeworld levels of the organisational context (Figure 43), in agreement with the findings of the studies in Chapters 7 and 8: these studies have indeed shown how the changes caused by the training at the organisational structure level affected the lifeworld level as well, ultimately resulting in a shift from a technology-centred mindset to a more user-centred one. In addition, the study in Chapter 8 has shown that participants have introduced the techniques proposed in the training both in their own daily practice and to their colleagues, extending the impact of the training (and

therefore of the framework) also to the work practice level of the organisation (Figure 43). I argue that this amplified effect can be due to that company being more accustomed to Agile, to the point that in its case the training only addressed UCD techniques: as suggested in section 8.6, a deeply rooted Agile mindset, characterised by self-organisation of teams and autonomy of developers, is likely to have favoured the appropriation of the design techniques that participants found more useful with respect to their work practice. Organisation and culture are among well-known, challenging themes in Agile (Gregory et al., 2016): a company that has already gone through the related cultural changes is thus likely to be more apt to also adopt UCD.

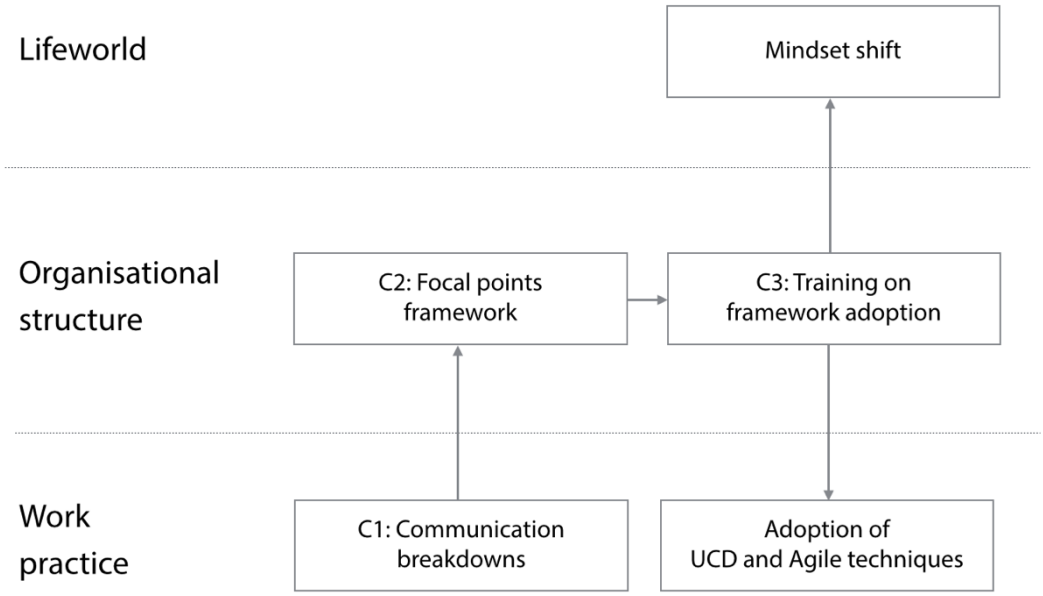


Figure 43 - Thesis contributions and their impact in the organisation

9.1 Limitations

For what concerns the generalisability of presented results, I remark that this thesis relied on a qualitative research approach because of its suitability to explore the situated practice of software engineering (Ferreira et al., 2011), emphasising practitioners' point of view (Abelein et al., 2013) and contributing a richer, more informative understanding of software development as a social and cooperative process (Dittrich et al., 2007; Sharp et al., 2016). Collected qualitative data are intrinsically specific to the related studies organisation, and are influenced by its situatedness and contingencies (Abelein et al., 2013). Nonetheless, it is possible to generalise the insights obtained through qualitative approaches beyond the single case (Sharp et al., 2016): therefore, this piece of work contributes to the body of empirical and action

research advocated by several authors (Brhel et al., 2015; Da Silva et al., 2011; Dybå and Dingsøyr, 2008) on the integration of UCD and Agile.

A final remark concerns how this research has been influenced by the specific companies that were willing to participate to studies, engaging in the potentially uncomfortable situation of exposing their organisational dynamics and practices to an external observer. While there is a broad consensus in literature on the paramount relevance of involving practitioners in research on the understanding and improvement of their practice (e.g. Dittrich et al., 2008; Grinter, 2003; Sohaib and Khan, 2010), it is anyway hard to achieve such involvement, due to the frequent difficulties and resistances in winning the sponsorship of companies. I second the call to academia to identify more actionable ways to prove the advantages of a mutual collaboration with industry, and to take the responsibility of translating its findings in a way that is accessible to industry (Ardito et al., 2014): indeed, some knowledge and technology transfer initiatives, such as the EIT Digital programme that complemented my doctoral studies, go in this direction. I also advocate, however, more openness in the establishment of an ongoing dialogue between industry and academia, ideally supported by suitable systemic policies.

9.2 Future work

There are several avenues for future work that can be pursued. First of all, the framework of focal points can be applied in other settings. One possibility would be the exploration of its applicability in organisational contexts different from those researched in this thesis: for instance, non-Italian companies, to investigate if the influence of different markets on the organisational structure is relevant, or large corporates, to investigate whether and how the framework needs to be revised in order to accommodate a more structured environment. In addition, rather than just supporting the integration of UCD and Agile in a company that has already decided to enact this kind of process, the framework can also be used as a diagnostic tool to assess the readiness of a company to adopt the integration in the first place.

Another direction for future endeavours is consolidating the evaluation of the impact of the framework on product quality besides on the organisational context. A first step in this direction has been described in section 7.5.1, but an important contribution in this sense would come from a greater engagement of customers in the adoption of UCD and Agile: this would facilitate access to users and to their involvement, easing requirement collection and product evaluation. To this end, the research community should undertake an effort to find ways of communicating more clearly the benefits that user involvement can have on the quality of products and work processes, for example by elaborating a set of actionable measures that can more objectively

assess this positive impact (Mao et al., 2005) or a set of less resource-intensive practices to put user involvement in place (Øvad and Larsen, 2016a).

10 Appendix

This appendix contains the interview guides for the studies described in this thesis. Since all of the studies were performed in Italian-speaking companies, all of the following material has been translated from Italian except for the interview guide in section 10.1.1, because the researcher who performed the study was foreign.

10.1 Smart Campus

This section presents the interview guides for the two interview studies analysed retrospectively (see section 4.1.2): the first one concerned the perception of user involvement, while the second one focused on project documentation. The two interview studies were carried out by two different researchers, whereas I analysed the results.

10.1.1 First interview study – Perception of user involvement

Interviewees: Stakeholder (Students, Founder) / associate (Software Engineer, Designer, HCI researcher, Project Manager)

10.1.1.1 Contents

0) Welcome address and introduction

Possible locations: accustomed work area, University of Trento. Provision of snacks

1) Getting started and general project overview + Evaluation (ca. 15 min)

The stakeholder gets a short version of the questions (5 min)

Warm-up and connection to Smart Campus (since when, how often, objectives, etc.)

Usual actions concerning the project (tools, meetings, created artifacts, project culture/collaboration climate etc.)

Evaluation of the project course so far

2) Participatory Design (PD) Process + Evaluation (ca. 25 min)

Possibilities and objectives of the PD process

Experiences (positive and negative), relationship to other stakeholders or associates

Evaluation: Impact and contribution of the PD on Smart Campus project and the own work

3) Prospects and Conclusion (ca. 5 min)

Future actions

Possibilities of improvement

4) Debriefing

10.1.1.2 Interview guide

(0)Welcome address and introduction

First of all, I want to thank you for your availability (disposition) to participate in this evaluation. This interview is going to be about the Smart Campus project. The focus will be on the realization of the participatory design process.

During the interview it is all about your personal impression, it will not influence your performance at university. That means there are no right or wrong answers and I´m interested in your honest opinion. It´s important to me, that you feel free to say what you like but also what you don´t like about the participation in the Smart Campus project so far. As a volunteer to this interview, you can end it anytime you want.

Furthermore I would like to record the interview to simplify the recapitulation, but the data will be treated in strict confidence and can only be used anonymously for research purposes. If you do agree, please sign this privacy agreement. Overall the interview will last about 30/45 minutes. Do you have any further questions?

1) Getting started and general overview + Evaluation (15 min)

To get into the topic: How would you describe the Smart Campus project to a friend of yours, who doesn´t know anything about it?

What are the main arguments?

Is PD mentioned? In which way?

What kind of terms, artifacts, tools and work parts are referred?

What are the main objectives of the project/your work in your opinion?

You are part of the Smart Campus Lab/You're a user of the Smart Campus Apps. Could you please sum up which connection do you have to the Smart Campus project?

Since when have you been participant/associate/researcher/stakeholder?

How often are you on duty for the project/ do you use Smart Campus?

How does a usual day with Smart Campus look like?

What are typical actions? In which order?

Which tools are normally used? (Details)

Are there any meetings? Regularly?

How does the coordination work?

How would you describe the project culture/climate?

Is there an awareness of other people´s actions?

Are you satisfied with the proceeding of the Smart Campus project so far? Why?/Why not?

Any positive/negative Experiences? (Examples? Misunderstandings? Best practices?)

What could be improved?

2) Participatory Design Process + Evaluation (ca. 25 min)

I´ve already mentioned that the interview focuses on the participatory design process in Smart Campus. So, let´s talk about your personal view of it…

In general terms: What does "participatory design" mean to you?

Which steps are included?

What kind of process is there?

What are the objectives?

Where do you see possibilities?

What are the pros/cons?

And in terms of Smart Campus: Could you please describe shortly the PD actions that were realized so far?

How have you participated? How often? (Details)

What are your positive/negative experiences? (Examples? Misunderstandings? Best practices?)

Questions for researchers

Elaboration and relationship to the other parties

How did you establish relationships with communities, participants, or users and stakeholders prior to commencing participatory research?

Were there certain participants selected or invited to take part over others?

Who determined the research context and the setting it took place in, or what research questions are formed?

What agendas, skills and assumptions did the participants bring to Smart Campus participatory design?

How would you assess the amount of participation by the stakeholders?

Evaluation

How do you reflect upon the complicated processes of participation and engagement while working with groups?

How are researchers and participants given space to document and reflect upon the activities they perform and how does this inform the research or design process throughout?

Furthermore, is it possible to document participatory work along the way without skewing the process itself?

As a last point: In your opinion, did the participatory design process create added value, which is sustainable?

What are the practical challenges of leaving a legacy of a participatory project?

Is sustainability or legacy always a positive outcome of participatory research?

Questions for other associates

Elaboration and relationship to the other parties

How do you assess the relationships with communities, participants, or users and stakeholders?

Were there certain participants selected or invited to take part over others?

Who determined the research context and the setting it took place in?

What agendas, skills and assumptions did you bring to Smart Campus participatory design?

How would you assess the amount of participation by the students?

Evaluation

Did you have the feeling to have access to all relevant information?

How do you assess the possibility for taking an independent position on a certain problem?

What do you think about the participatory development methods which were used (where, how, with whom, etc.)?

Did you feel as if there were any room for alternative technical and/or organizational arrangements?

Is it possible to document participatory work along the way without skewing the process itself?

As a last point: In your opinion, did the participatory design process created added value, which is sustainable?

What are the practical challenges of leaving a legacy of a participatory project?

Is sustainability or legacy always a positive outcome of participatory research?

Questions for stakeholders

Elaboration and relationship to the other parties

How did you experience the possibilities to participate in Smart Campus so far?

How do you assess the relationships with communities, participants, or users and stakeholders?

Who determined the research context and the setting it took place in?

What agendas, skills and assumptions did you bring to Smart Campus participatory design?

How would you assess your amount of participation to the project?

And how did the decision-making work? Did you have the impression that you have been part of it?

Evaluation

Did you have the feeling to have access to all relevant information?

How do you assess the possibility for taking an independent position on a certain problem?

What do you think about the participatory development methods which were used (where, how, with whom, etc.)?

Did you feel as if there were any room for alternative technical and/or organizational arrangements?

Is it possible to document participatory work along the way without skewing the process itself?

As a last point: In your opinion, did the participatory design process created added value, which is sustainable?

What are the practical challenges of leaving a legacy of a participatory project?

Is sustainability or legacy always a positive outcome of participatory research?

3) Prospects and Conclusion (ca. 5 min)

What kind of participatory design measures would you see in the future work of Smart Campus?

Are there any improvements in the PD process that should be considered?

Is there anything else you would like to add?

4) Debriefing

Ok. That's it. Thank you very much for your participation!

10.1.2 Second interview study – Project documentation

1. How do you normally document textually your development process? How do you pass this information onto the other developers?

2. Are there other ways that you have been documenting the development process? Pictures? Sketches? Other? If so, what have you been using? How do you share them with others?

3. Do you use GitHub for this kind of documentation? Any other websites?

4. What about the Smart Campus forum?

5. a)If yes, how do you use it?

   b)If no, why don't you use it? Do you think it would make sense to use it for this purpose?

6. How do you use the forum at the moment?

10.2 H-umus

This section presents the interview guide for the interviews that I carried out at H-umus (see section 5.1.1). I designed the interview guide itself, performed and transcribed the interviews, and analysed them.

Hi, I am Silvia and I am a PhD student at the University of Trento. My research concerns the way Agile development and user-centred design are integrated in companies. With this interview, I am interested in understanding how this integration happens here at H-umus, and how you perceive it. I'd like to point out, before we start, that the questions I'll ask are only meant to guide our conversation: therefore, if you feel like adding more details, or that there is something important to add, please go ahead.

1. What is your name? What is your role in H-umus? Can you tell me a bit about your background? in which projects are you involved?
2. Tell me about a typical week in your job: What kind of tasks do you usually perform? Are you always on site?
3. Are you supported by any tools or artefacts? If so, can you show them to me and tell me more about how you use them?

For designers and developers:

4. Can you please show me the task you are currently working on and talk me through it?
5. Do you follow any specific design / development methodology in your activities? If so, can you tell me more about it?
6. Do you usually coordinate with other colleagues? If so, with whom? How? Do you use any tools or artefacts to do it?
7. As a designer / developer, are you involved in development / design activities? If so, how?
8. Do you interact with any customer? If so, how?
9. Do you also interact with the users of the software? If so, how and how frequently? If not, what is the reason?

For project managers:

10. Can you please tell me more about the projects you are managing? I am interested in aspects such as the activities you are responsible for, how frequently do you interact with the customer and how, which is the process for engaging the customer, etc.
11. Are the projects you are managing usually long-term or short-term? Do you manage several of them at the same time or are you dedicated to a single project or customer?
12. How does communication with the customer occur? Do you use any devices to support it?
13. Do you also interact with the users of the software? If so, how and how frequently? If not, what is the reason?

14. How do you interact with designers and developers? How do you share with them the customer's requirements, and how do you follow up throughout the project lifecycle?

10.3 Delta Informatica

This section presents the interview guide for the interviews carried out at Delta Informatica. I will first present the guide for the preliminary interviews designed and performed by other researchers (see section 7.2.1), and then the guide for the final evaluation interviews, which I designed, transcribed, and analysed while a fellow researcher actually performed them (see section 7.5.2).

10.3.1 Preliminary interviews

1. When did you start working on PRESTO?
2. Do you work as a team? How many people are involved?
3. Who set the goals of the project, and how? How is the project coordinated?
4. Let us talk about your daily work practices. How do you usually carry out a task that is assigned to you?
5. If you need to coordinate with a colleague, how do you do it? Do you use any kind of supporting device, either tangible or digital?
6. Is any specific development methodology in use in PRESTO? If so, can you describe this methodology? Do you know why it was chosen?
7. Which tools did you use to implement the software? And to coordinate the work of the team?
8. Are there any differences between the current design of PRESTO and the design envisioned at the beginning of the project? If so, which are the main differences? What are the reasons that led to these modifications?
9. How was the development process documented?
10. Who is the user envisioned for PRESTO? Is this the same user that was envisioned at the beginning of the project? If not, why was it re-defined?
11. In your opinion, what should be the role of the user with respect to the design of the software? And with respect to its implementation?
12. Would you say that the needs and expectations of the user were accounted for in PRESTO? If so, how was this contribution incorporated in the design and development process?
13. Do you think PRESTO requires any changes to further accommodate the needs of final users? If so, which ones?

10.3.2 Evaluation interviews

Warm-up

What was your role in PRESTO? How long have you been participating to the project? *(in short)*

What did you know about user-centred design before joining the project? Were you interested in this topic? *(probe whether attitude was positive or negative)*

What did you know about Agile development before joining the project? Were you interested in this topic? *(probe whether attitude was positive or negative)*

Before the intervention

Let us talk about the work practice in PRESTO before you started the set of workshops.

Did the development team stay the same over time? Did you use any tools or practices to coordinate? If so, which ones?

Was anybody in charge of the design? How did this person coordinate with develoeprs? *(sensitive question: we know the design was done by University personnel)*

Let us now move to talking about the training. In general, what is your overall opinion about it?

Design training

(show design probes) During the design workshops you have met the following techniques, which you can see summarised in the probes: communication networks, personas and two-level personas, task analysis through use-case diagrams, interaction metaphors, prototypes, quality criteria and planning poker, functional and non-functional requirements, building of a shared terminology. Among all these techniques, which one was most useful in your opinion and why? And which one was the least useful and why?

Do you think the design training changed your understanding of the project? If so, how?

Do you think the design training changed the dynamics within the team? If so, how?

Are there any aspects of design and of the user you would have liked to discuss more thoroughly? If so, which ones?

Development training

During the development workshops you have met the following techniques: Scrum board, backlog, daily meetings, time-boxed sprints, periodical sprint reviews. Among all these techniques, which one was most useful in your opinion and why? And which one was the least useful and why?

Are there any aspects of the development process you would have liked to discuss more thoroughly? If so, which ones?

Do you think that the outcomes of the design training have influenced the development? If so, how?

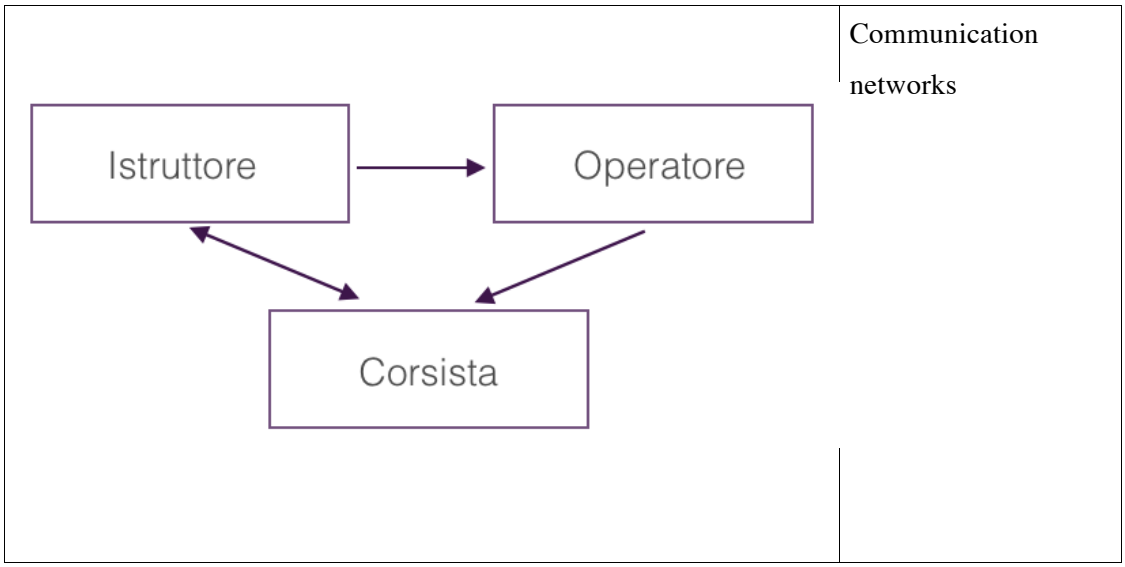
Do you think that the development training influenced your work practice? If so, how?

Do you think that the t development training influenced the coordination within the team? If so, how?

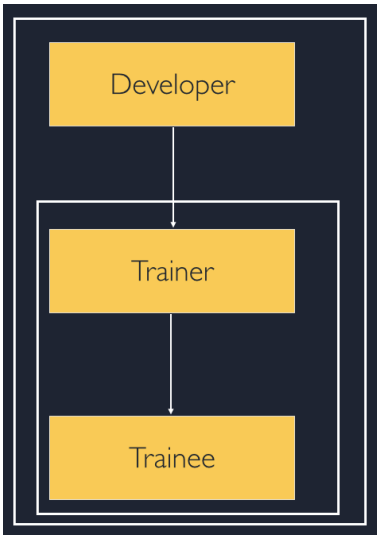
Do you think that the development training influenced the outcome of the project? If so, how?

In general, do you have any suggestions to improve the process followed through the design and development training?

Design probes



Personas

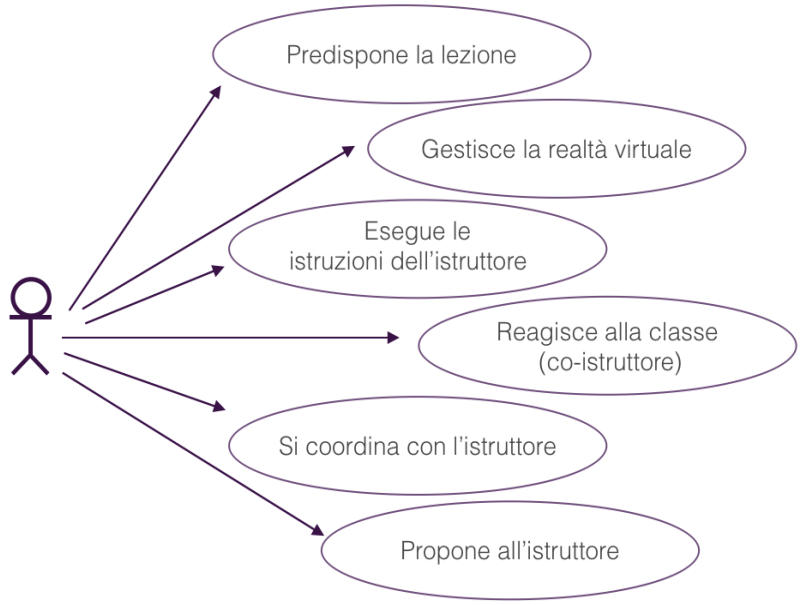


Two-level personas

Developer-mediated	Lo sviluppatore ipotizza le necessità dell'utente senza rapporto diretto con esso
Customer-centred	Il cliente media completamente le necessità dell'utente e la relazione con esso
Informative	Gli utenti forniscono e/o ricevono informazioni
Consultative	Gli utenti commentano un insieme predefinito di funzionalità
Participative	Gli utenti influenzano attivamente la progettazione del sistema

Extent of user involvement

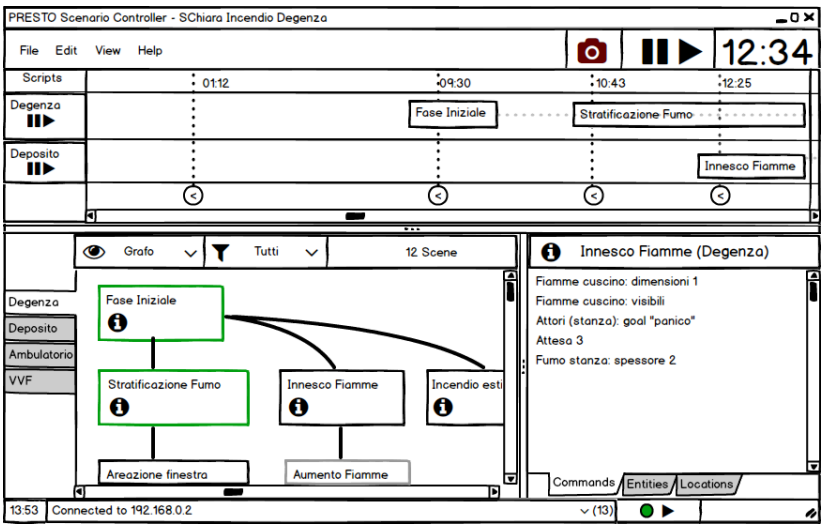
Use case diagrams



Interaction metaphors

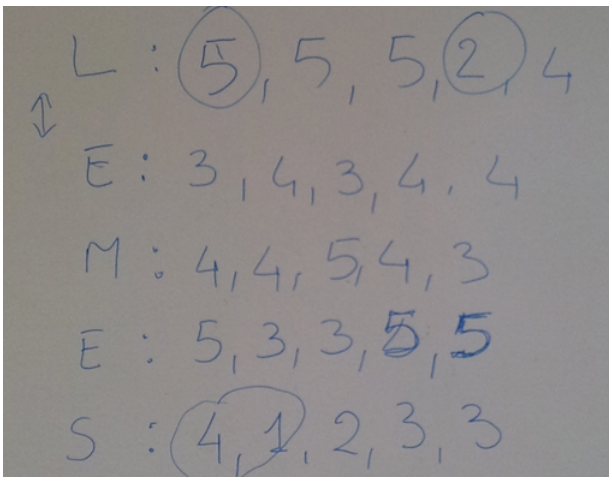
Metafora	Pro	Contro
Telecomando	Opzioni predefinite e limitate, dispositivo dedicato, efficienza	Espressività ridotta, non fornisce visione d'insieme
Prezi	Visione d'insieme, navigazione non necessariamente sequenziale, opzioni limitate	Espressività ridotta, facile degrado dell'usabilità
~~Libro interattivo~~	~~Opzioni predefinite e limitate, navigazione sequenziale~~	~~Espressività ridotta, non fattibile~~
~~Pannello interruttori~~	~~Visione d'insieme, grande espressività~~	~~Scarsa usabilità~~

Low-fidelity prototypes



Dimensione	Definizione
Learnability	Facilità di apprendimento del modo in cui usare l'interfaccia
Efficiency	Efficienza nell'utilizzo
Memorability	Facilità nel ricordare come utilizzare l'interfaccia
Errors	Sicurezza e robustezza
Satisfaction	Soddisfazione nell'utilizzo

Quality criteria



Planning poker

A	B	C
1	Sistema	
1,1	Connessione/Disconnessione	si collega alla VR (PRESTO)
1,2	Caricamento di uno script	da una lista di nomi data dal server
1,3	Visualizzazione dettagli script	descrizione, attori, location, etc.
1,4	Visualizzazione stato del sistema	connessione, script caricato, etc.
1,5	Messaggi di allerta	al verificarsi di azioni di estemporanee (es. disconnession
1,6	Override goal degli agenti	sottomette goal agli agenti senza passare dalla sceneggia
2	Time control	
2,1	Avvio/Pausa di uno script	gli avanzamenti automatici dello script sono inibiti
2,2	Avvio/Pausa di tutti gli script (globale)	gli avanzamenti automatici di tutti gli script sono inibiti
2,3	Avvio/Pausa della VR	oltre a bloccare il tempo degli script, mette in pausa la VR
2,4	Visualizzazione tempo di uno script	il tempo trascorso per un singolo script
2,5	Visualizzazione tempo di tutti gli script (globale)	il tempo trascorso nell'esecuzione di tutti gli script
2,6	Visualizzazione tempo VR	il tempo trascorso nella VR
2,7	Creazione manuale di un punto di ripristino globale	salva stato VR e di tutti gli script attuali (checkpoint global
2,8	Stop della sceneggiatura	ripristina lo stato iniziale e scarica gli script
3	Timeline	
3,1	Visualizzazione delle scene in ordine di esecuzione	lista
3,2	Visualizzazione delle scene raggruppate per script	lista di liste (solo script di alto livello)
3,3	Visualizzazione della dimensione temporale	collocate su di una linea temporale (comprende quindi sp
3,4	Visualizzazione dei punti di ripristino per script	includerebbe anche quelli manuali
3,5	Visualizzazione dei punti di ripristino globali	quelli manuali
3,6	Attivazione di un punto di ripristino	riporta indietro il tempo dello script, il suo stato e la VR
3,7	Attivazione di un punto di ripristino globale	riporta indietro il tempo di tutti gli script, il loro stato e tutta
3,8	Visualizzazione durata scene	a seconda del tempo che ci mettono i comandi ad essere
3,9	Visualizzazione scene ancora in corso	se ancora ci sono comandi da essere eseguiti

Functional and non-functional requirements

	Shared terminology

10.4 eMaze

This section presents the interview guide for the interviews carried out at eMaze. I will first present the guide for the preliminary interviews (see section 8.2), and then the guide for the final evaluation interviews, which I designed, transcribed, and analysed while a fellow researcher actually performed them (see section 8.4).

10.4.1 Preliminary interviews

Hi, I am Silvia and I am a PhD student at the University of Trento. My research concerns the way Agile development and user-centred design are integrated in companies. With this interview, I am interested in understanding how this integration happens here at eMaze, and how you perceive it. I'd like to point out, before we start, that the questions I'll ask are only meant to guide our conversation: therefore, if you feel like adding more details, or that there is something important to add, please go ahead.

1. What is your name? What is your role in eMaze? Can you tell me a bit about your background? in which projects are you involved?
2. Can you tell me how the company is structured from your point of view?
3. Tell me about a typical week in your job: What kind of tasks do you usually perform? Are you always on site?
4. Are you supported by any tools or artefacts? If so, can you show them to me and tell me more about how you use them?
5. Can you please show me the task you are currently working on and talk me through it?
6. Do you follow any specific design / development methodology in your activities? If so, can you tell me more about it?

7. Do you usually coordinate with other colleagues? If so, with whom? How? Do you use any tools or artefacts to do it?
8. How does communication with the customer occur? Do you use any devices to support it?
9. Do you also interact with the users of the software? If so, how and how frequently? If not, what is the reason?

10.4.2 Evaluation interview

Warm-up

What is your role in eMaze? How did you participate in the Validatore project? *(in short)*

What did you know about user-centred design before the workshops? Were you interested in this topic? Had this approach beenmentioned or adopted in the company? *(probe whether attitude was positive or negative)*

What did you know about Agile development before the workshops? Were you interested in this topic? Had this approach beenmentioned or adopted in the company? *(probe whether attitude was positive or negative)*

Before the intervention

Please describe briefly the goals of the Validatore project.

Let us now talk about the process that led the Validatore to its current state.

Who are its stakeholders? How do they influence the project? What was the role of users in the design and development of the Validatore?

Has the project evolved linearly, or as envisioned at first? Has the team stayed the same over time?

About the intervention

In general, what is your overall opinion about this training?

(show design probes) During the design workshops you have met the following techniques, which you can see summarised in the probes: building of a shared terminology, communication networks, task analysis through use-case diagrams, user research, personas, scenarios, storyboards, quality criteria and planning poker, interaction metaphors, wireframing, prototypes, design principles, user evaluation. Among all these techniques, which one was most useful in your opinion and why? And which one was the least useful and why?

Did you have the chance of applying any of these techniques in your daily work practice in the last months? If so, which ones?

Did you share your experience with the colleagues who did not participate in the training? If so, how?

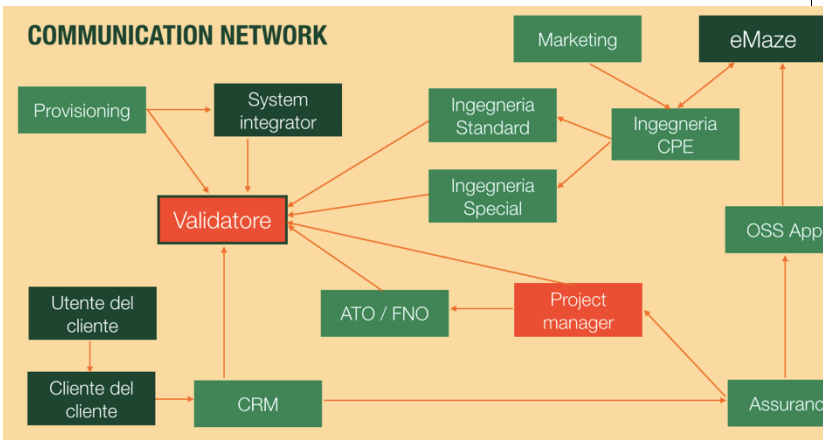
Do you think the training influenced your daily way of working? If so, how?

Do you think the training influenced your attitude towards software projects? If so, how? *(probe attitude towards a user-centred mindset)*

Are there any aspects of design and of the user you would have liked to discuss more thoroughly? If so, which ones?

In general, do you have any suggestions to improve the process followed through the design and development training?

Probes used

	Communication network

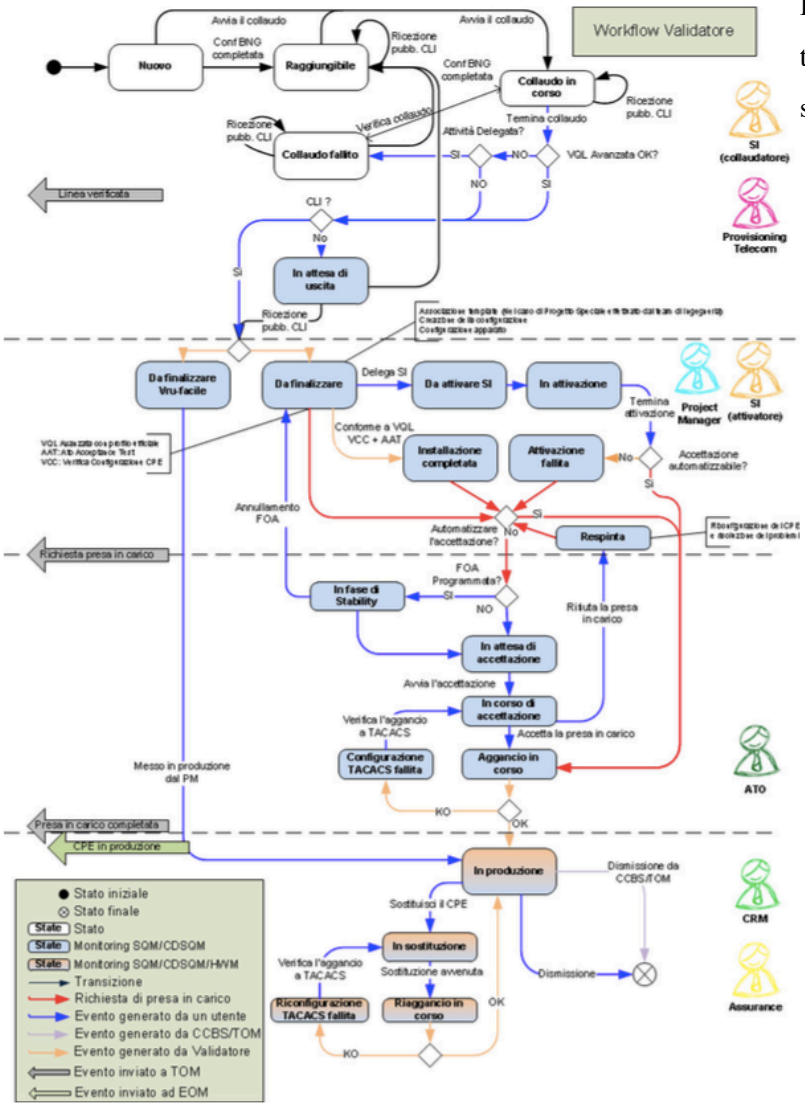
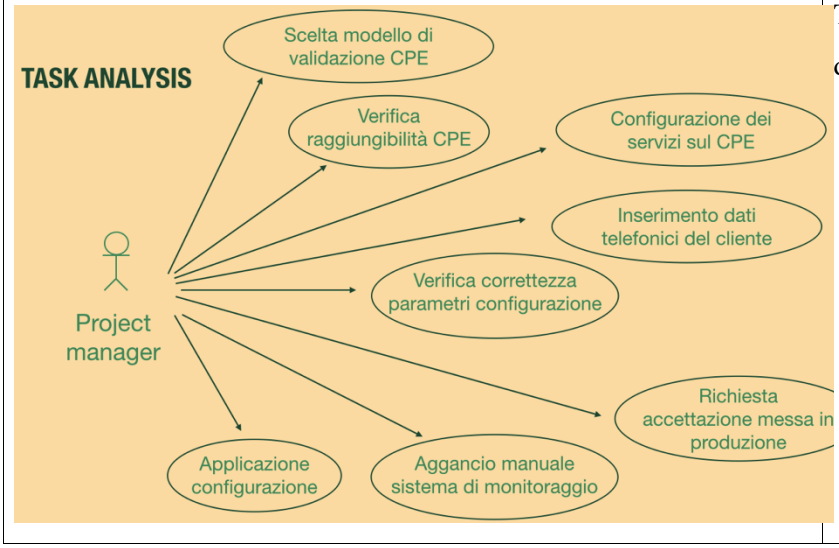


Figura 2.2: Workflow di un CPE.

Building a shared terminology from the software manual



Task analysis / Use case diagrams

User research

Personas

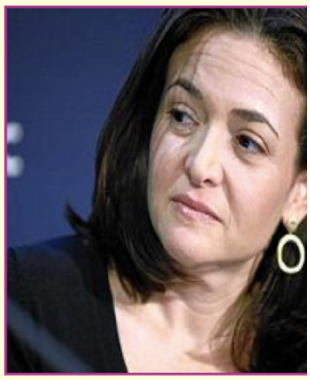
PERSONAS - LUCREZIA

PM della indiretta

Forti competenze tecniche, commerciali e informatiche; molto motivata

Utilizza il Validatore anche per svolgere compiti complessi di grande rilevanza commerciale

Si aspetta un'interfaccia sicura, efficiente e che preveda opzioni avanzate

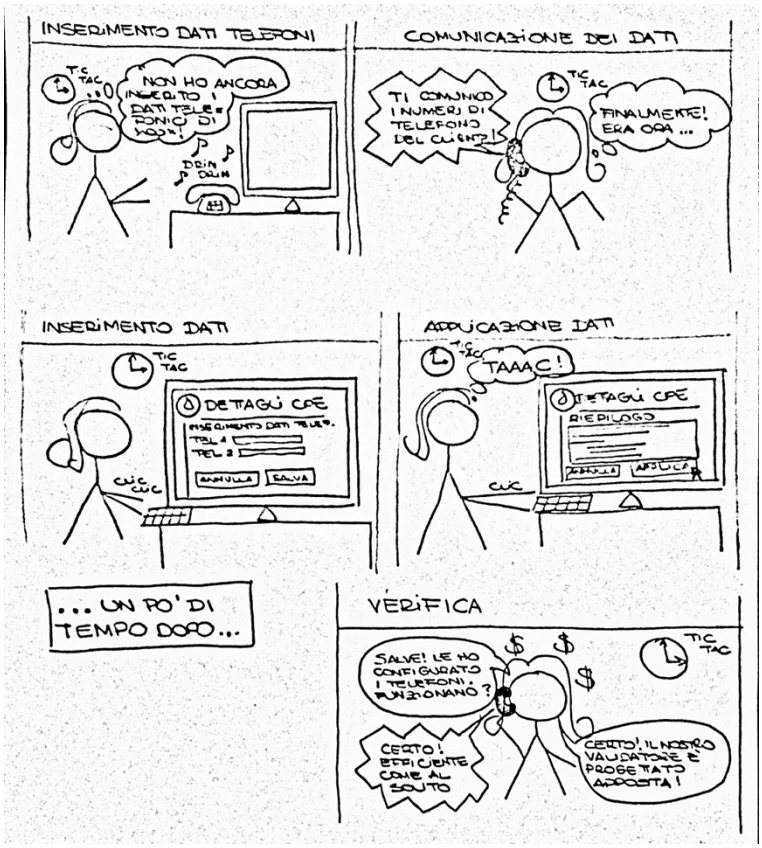


Scenarios

SCELTA DI UN MODELLO DI VALIDAZIONE

Lucrezia deve leggere dal contratto che cosa ha venduto al cliente e quindi creare una cor
coerente.
Apre il Validatore, seleziona il CPE che deve essere configurato e va alla sezione dove può
il modello di validazione.
Il sistema la assiste nella compilazione dei vari moduli, mostrandole le opzioni possibili in
scelte precedentemente effettuate.
Le è chiesto di selezionare i dati (come ad esempio il vendor, il modello, connettività voce
configurazione per ogni livello obbligatorio. Dopodiché seleziona i eventi/servizi aggiuntiv
ciò che è consentito. Infine il modello di validazione viene salvato.
Lucrezia deve poter avere visibilità dei livelli del modello di validazione creato.
E' importante che il sistema la assista in modo che possa completare le sue attività veloce
che le metta a disposizione strumenti avanzati per poter creare un modello di validazione
risponda alle esigenze ad hoc dei suoi clienti.

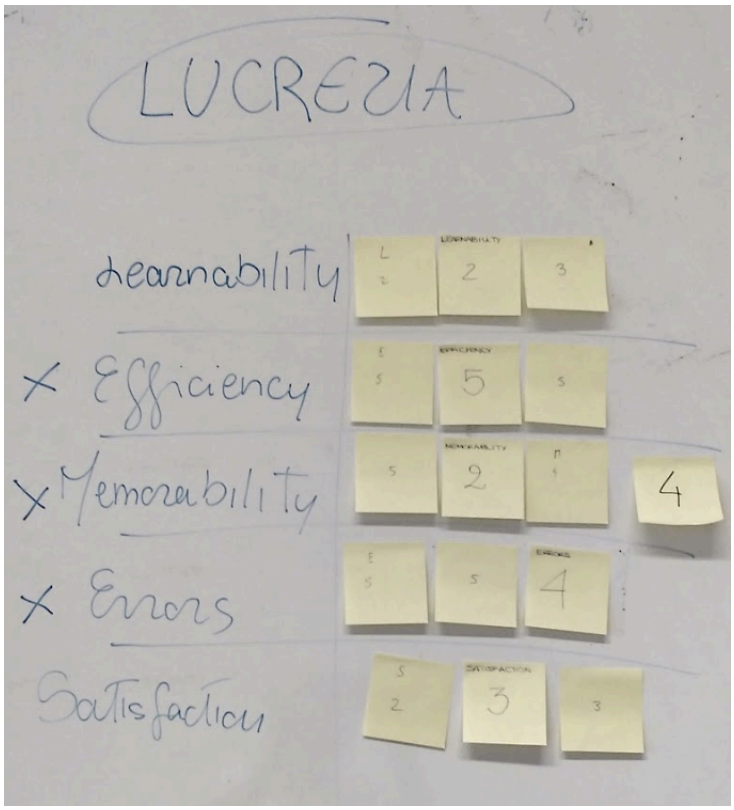
Storyboards



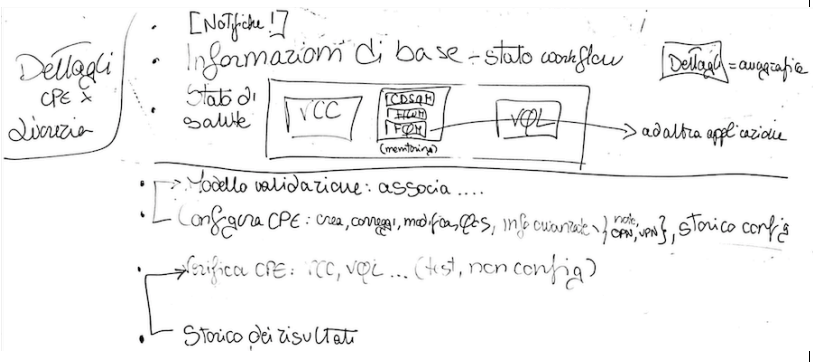
CRITERI DI QUALITÀ

Dimensione	Definizione	Pri
Learnability	Facilità di apprendimento del modo in cui usare l'interfaccia	
Efficiency	Efficienza nell'utilizzo	
Memorability	Facilità nel ricordare come utilizzare l'interfaccia	
Errors	Sicurezza e robustezza	
Satisfaction	Soddisfazione nell'utilizzo	

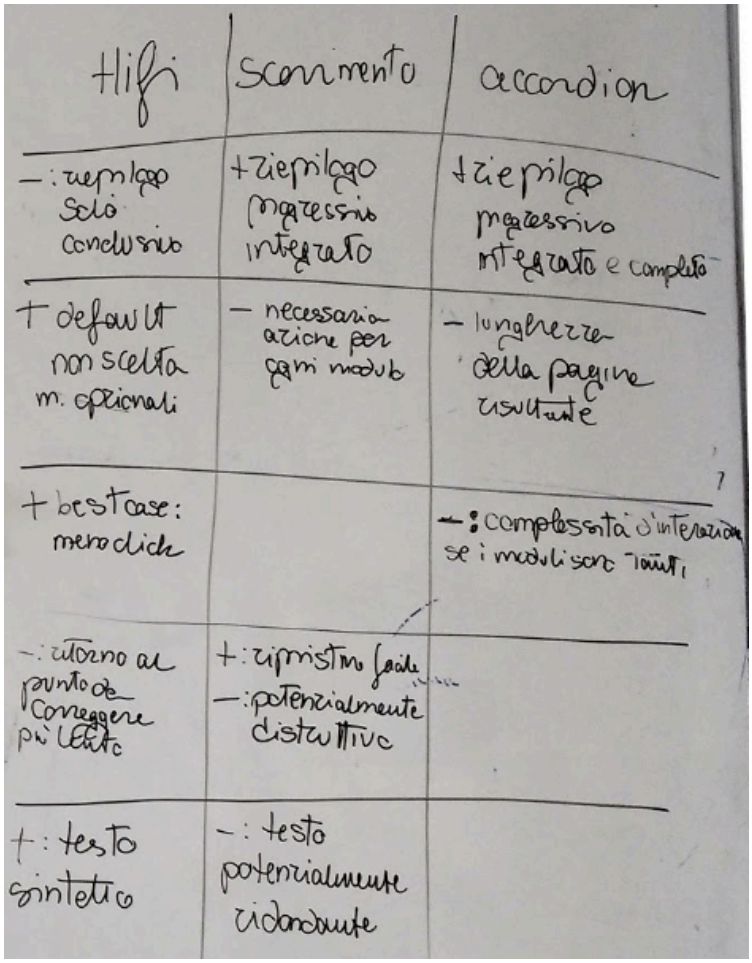
Quality criteria



Planning poker

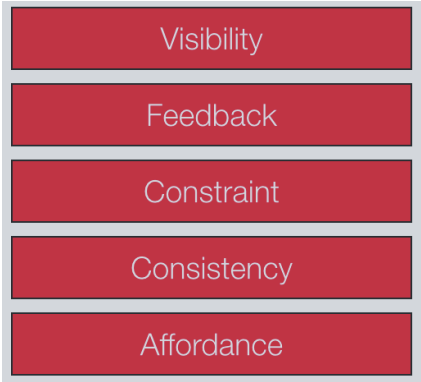


Wireframing



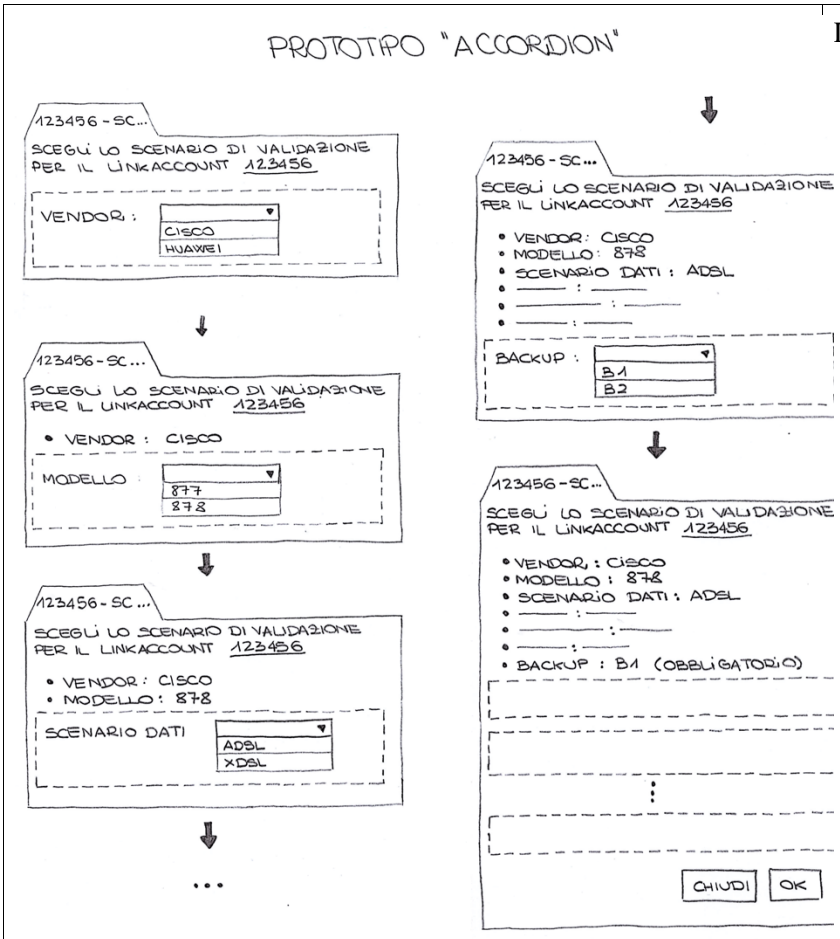
Hifi	Scorrimento	accordion
–: riepilogo solo conclusivo	+riepilogo progressivo integrato	+riepilogo progressivo integrato e completo
+ default non scelta m. opzionali	– necessaria azione per ogni modulo	– lunghezza della pagina risultante
+ best case: mero click		–: complessità d'interazione se i moduli sono tanti
–: ritorno al punto da correggere più lento	+: ripristino facile –: potenzialmente distruttivo	
+: testo sintetico	–: testo potenzialmente ridondante	

Interaction metaphors



Design principles

Low-fidelity prototypes



User evaluation

VALUTAZIONE EURISTICA (J. NIELSEN, 1994)

Visibility of system status	Recognition rather than recall
Match between system and the real world	Flexibility and efficiency of use
User control and freedom	Aesthetic and minimalist desig
Consistency and standards	Help users recognize, diagnos and recover from errors
Error prevention	Help and documentation

Bibliography

Abelein, U., Sharp, H., Paech, B., 2013. Does involving users in software development really influence system success? IEEE software 30, 17–23.

Agile Business Consortium, URL https://www.agilebusiness.org/

Agile Business Day, URL http://agilebusinessday.com/

Ambler, S.W., 2015a. Introduction to user stories. http://www.agilemodeling.com/artifacts/userStory.htm

Ambler, S.W., 2015b. Agile/Lean Documentation: Strategies for Agile Software Development, in: Http://Www.agilemodeling.com/Essays/agileDocumentation.htm

Ardito, C., Buono, P., Caivano, D., Costabile, M.F., Lanzilotti, R., 2014. Investigating and promoting UX practice in industry: An experimental study. International Journal of Human-Computer Studies 72, 542–551.

Atkinson, P., Hammersley, M., 1994. Ethnography and participant observation.

Balsamiq Mockups, URL https://balsamiq.com/products/mockups/

Barrett, M., Oborn, E., 2010. Boundary object use in cross-cultural software development teams. Human Relations 63, 1199–1221.

Baskerville, R.L., Wood-Harper, A.T., 1996. A critical perspective on action research as a method for information systems research. Journal of information Technology 11, 235–246.

Beck, K., 2001. Manifesto for Agile software development.

Beck, K., 2000. Extreme programming explained: embrace change. Addison-Wesley Professional.

Beyer, H., Holtzblatt, K., Baker, L., 2004. An agile customer-centered method: rapid contextual design, in: Conference on Extreme Programming and Agile Methods. Springer, pp. 50–59.

Bjerknes, G., Bratteteig, T., 1995. User participation and democracy: A discussion of Scandinavian research on system development. Scandinavian Journal of information systems 7, 1.

Bjørn, P., Bardram, J., Avram, G., Bannon, L., Boden, A., Redmiles, D., De Souza, C., Wulf, V., 2014. Global software development in a CSCW perspective, in: Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing. ACM, pp. 301–304.

Bjørn, P., Ngwenyama, O., 2009. Virtual team collaboration: building shared meaning, resolving breakdowns and creating translucence. Information Systems Journal 19, 227–253.

Blumer, H., 1954. What is wrong with social theory? American sociological review 19, 3–10.

Boden, A., Rosswog, F., Stevens, G., Wulf, V., 2014. Articulation spaces: bridging the gap between formal and informal coordination, in: Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing. ACM, pp. 1120–1130.

Boehm, B.W., 1984. Software engineering economics. IEEE transactions on Software Engineering 4–21.

Bordin, S., De Angeli, A., 2017. Inoculating an Agile Company with User-Centred Design: An Empirical Study, in: International Conference on Agile Software Development. Springer, pp. 235–242.

Bordin, S., De Angeli, A., 2016. Communication breakdowns in the integration of user-centred design and Agile development, in: Integrating User-Centred Design in Agile Development. Springer International Publishing, pp. 137–161.

Bordin, S., Menendez Blanco, Maria, De Angeli, Antonella, 2014. ViaggiaTrento: an application for collaborative sustainable mobility. EAI Endorsed Transactions on Ambient Systems 14.

Bornoe, N., Stage, J., 2014. Usability Engineering in the Wild: How Do Practitioners Integrate Usability Engineering, in: Software Development?. In International Conference on Human-Centred Software Engineering. Springer Berlin, Heidelberg, pp. 199–216.

Bowen, G.A., 2006. Grounded theory and sensitizing concepts. International journal of qualitative methods 5, 12–23.

Braun, V., Clarke, V., 2006. Using thematic analysis in psychology. Qualitative research in psychology 3, 77–101.

Brhel, M., Meth, H., Maedche, A., Werder, K., 2015. Exploring principles of user-centered agile software development: A literature review. Information and Software Technology 61, 163–181.

Brown, J.M., 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.

Brown, T., 2008. Design thinking. Harvard business review 86, 84.

Bruun, A., 2010. Training software developers in usability engineering: a literature review., in: Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending. Boundaries. ACM.

Cajander, Å., Lárusdóttir, M., Gulliksen, J., 2013. Existing but not explicit-The user perspective in scrum projects in practice, in: IFIP Conference on Human-Computer Interaction. Springer, pp. 762–779.

Cataldo, M., Herbsleb, J.D., 2008. Communication networks in geographically distributed software development, in: Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work. ACM, pp. 579–588.

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. Presented at the International Conference on Extreme Programming and Agile Processes in Software Engineering, Springer, Berlin, Heidelberg, pp. 143–153.

Cho, L., 2009. Adopting an Agile Culture A User Experience Team's Journey, in: Agile Conference. pp. 9–416.

Clark, H.H., Brennan, S.E., 1991. Grounding in communication. Perspectives on socially shared cognition 13, 127–149.

Cockton, G., Lárusdóttir, M., Gregory, P., Cajander, Å., 2016. Integrating User-Centred Design in Agile Development, in: Integrating User-Centred Design in Agile Development. Springer, pp. 1–46.

Cohn, M., 2004. User stories applied: For agile software development. Addison-Wesley Professional.

Cohn, M.L., Sim, S.E., Lee, C.P., 2009. What Counts as Software Process? Negotiating the Boundary of Software Work Through Artifacts and Conversation. Comput Supported Coop Work 18, 401.

Cooper, A., 2004. The inmates are running the asylum:[Why high-tech products drive us crazy and how to restore the sanity]. Sams Indianapolis, IN, USA:

Coram, M., Bohner, S., 2005. The impact of agile methods on software project management, in: 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems. IEEE, pp. 363–370.

Curtis, B., Kellner, M.I., Over, J., 1992. Process Modeling. Commun. ACM 35, 75–90.

Da Silva, T.S., Martin, A., Maurer, F., Silveira, M., 2011. User-centered design and agile methods: a systematic review, in: Agile Conference (AGILE), 2011. IEEE, pp. 77–86.

Da Silva, T.S., Silveira, F.F., Silveira, M.S., Hellmann, T., Maurer, F., 2015. A systematic mapping on agile UCD across the major agile and HCI conferences, in: International Conference on Computational Science and Its Applications. Springer, pp. 86–100.

Damodaran, L., 1996. User involvement in the systems design process-a practical guide for users. Behaviour & information technology 15, 363–377.

De Angeli, A., Bordin, S., Blanco, M.M., 2014a. Infrastructuring participatory development in information technology, in: Proceedings of the 13th Participatory Design Conference: Research Papers-Volume 1. ACM, pp. 11–20.

De Angeli, A., Bordin, S., Menéndez Blanco, M., 2014b. Reflections over a socio-technical infrastructuring effort, in: Proceedings of 2nd Workshop on Cultures of Participation in the Digital Age, CoPDA.

De Angeli, A., Hartmann, J., Sutcliffe, A., 2009. The effect of brand on the evaluation of websites. Human-Computer Interaction–INTERACT 2009 638–651.

Dittmar, A., Hensch, M., 2015. Two-Level Personas for Nested Design Spaces, in: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, pp. 3265–3274.

Dittrich, Y., John, M., Singer, J., Tessem, B., 2007. For the special issue on qualitative software engineering research. Information and software technology 49, 531–539.

Dittrich, Y., Randall, D.W., Singer, J., 2009. Software engineering as cooperative work. Computer Supported Cooperative Work (CSCW) 18, 393.

Dittrich, Y., Rönkkö, K., Eriksson, J., Hansson, C., Lindeberg, O., 2008. Cooperative method development. Empirical Software Engineering 13, 231–260.

Dourish, P., Bellotti, V., 1992. Awareness and coordination in shared workspaces, in: Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work. ACM. pp. 107–114.

Dybå, T., Dingsøyr, T., 2008. Empirical studies of agile software development: A systematic review. Information and software technology 50, 833–859.

Dybå, T., Prikladnicki, R., Rönkkö, K., Seaman, C., Sillito, J., 2011. Qualitative research in software engineering. Empirical Software Engineering 16, 425–429.

Eriksson, E., Cajander, Å., Gulliksen, J., 2009. Hello world!–experiencing usability methods without usability expertise, in: IFIP Conference on Human-Computer Interaction. Springer, pp. 550–565.

Ferreira, J., 2008. Interaction Design and Agile Development: a Real World Perspective.

Ferreira, J., Sharp, H., Robinson, H., 2012. Agile development and user experience design integration as an ongoing achievement in practice, in: Agile Conference (AGILE), 2012. IEEE, pp. 11–20.

Ferreira, J., Sharp, H., Robinson, H., 2011. User experience design and agile development: managing cooperation through articulation work. Softw: Pract. Exper. 41, 963–974.

Flick, U., 2014. An introduction to qualitative research. Sage.

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.

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.

Gothelf, J., 2013. Lean UX: Applying Lean principles to improve user experience. O'Reilly Media, Inc.

Gray, D., Brown, S., Macanufo, J., 2010. Gamestorming: A playbook for innovators, rulebreakers, and changemakers. O'Reilly Media, Inc.

Gregory, J., 2003. Scandinavian approaches to participatory design. International Journal of Engineering Education 19, 62–74.

Gregory, P., Barroca, L., Sharp, H., Deshpande, A., Taylor, K., 2016. The challenges that challenge: Engaging with agile practitioners' concerns. Information and Software Technology 77, 92–104.

Grenning, J., 2002. Planning poker or how to avoid analysis paralysis while release planning. Hawthorn Woods: Renaissance Software Consulting 3.

Grinter, R.E., 2003. Recomposition: Coordinating a web of software dependencies, in: Computer Supported Cooperative Work (CSCW). pp. 297–327.

Gross, T., Stary, C., Totter, A., 2005. User-Centered Awareness in Computer-Supported Cooperative Work-Systems: Structured Embedding of Findings from Social Sciences. International Journal of Human–Computer Interaction 18, 323–360.

Guðjónsdóttir, R., Lindquist, S., 2008. Personas and scenarios: Design tool or a communication device?, in: Proceedings of the 8th International Conference on the Design of Cooperative Systems. pp. 165–176.

Gutwin, C., Penner, R., Schneider, K., 2004. Group awareness in distributed software development, in: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work. ACM, pp. 72–81.

Guzzi, A., Bacchelli, A., Riche, Y., van Deursen, A., 2015. Supporting developers' coordination in the ide, in: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing. ACM, pp. 518–532.

Häkli, A., 2005. Introducing user-centered design in a small-size software development organization. Helsinki University of Technology, Helsinki.

Halverson, C.A., Ellis, J.B., Danis, C., Kellogg, W.A., 2006. Designing task visualizations to support the coordination of work in software development, in: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work. ACM, pp. 39–48.

Hasnain, E., Hall, T., Shepperd, M., 2013. Using experimental games to understand communication and trust in Agile software teams, in: 2013 6th International Workshop

on Cooperative and Human Aspects of Software Engineering (CHASE). Presented at the 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), pp. 117–120.

Hill, J., 2007. Empowering teams: the Scrum Master's role.

Hoda, R., Noble, J., Marshall, S., 2011. The impact of inadequate customer collaboration on self-organizing Agile teams. Information and Software Technology 53, 521–534.

Hussain, Z., Milchrahm, H., Shahzad, S., Slany, W., Tscheligi, M., Wolkerstorfer, P., 2009a. Integration of extreme programming and user-centered design: Lessons learned, in: International Conference on Agile Processes and Extreme Programming in Software Engineering. Springer, pp. 174–179.

Hussain, Z., Slany, W., Holzinger, A., 2009b. Current state of agile user-centered design: A survey, in: Symposium of the Austrian HCI and Usability Engineering Group. Springer, pp. 416–427.

Iivari, J., Iivari, N., 2011. Varieties of user-centredness: an analysis of four systems development methods. Information Systems Journal 21, 125–153.

Jia, Y., Lárusdóttir, M., Cajander, Å., 2012. The Usage of Usability Techniques in Scrum Projects, in: Human-Centered Software Engineering. Presented at the International Conference on Human-Centred Software Engineering, Springer, Berlin, Heidelberg, pp. 331–341.

Jurca, G., Hellmann, T.D., 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.

Kane, D., 2003. Finding a place for discount usability engineering in agile development: throwing down the gauntlet, in: Agile Development Conference. IEEE, pp. 40–46.

Kollmann, J., Sharp, H., Blandford, A., 2009. The Importance of Identity and Vision to User Experience Designers on Agile Projects, in: 2009 Agile Conference. Presented at the 2009 Agile Conference, pp. 11–18.

Kuczynski, L., Daly, K., 2003. Qualitative methods for inductive (theory-generating) research. Handbook of dynamics in parent-child relations 373–392.

Kujala, S., 2003. User involvement: a review of the benefits and challenges. Behaviour & information technology 22, 1–16.

Kumana, D., Dickinson, J., 2014. Agile and user-centred design.

Kuusinen, K., Gregory, P., Sharp, H., Barroca, L., 2016. Strategies for doing Agile in a non-Agile Environment, in: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ACM, p. 5.

Kuutti, K., Bannon, L.J., 2014. The turn to practice in HCI: towards a research agenda, in: Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems. ACM, pp. 3543–3552.

Laanti, M., Similä, J., Abrahamsson, P., 2013. Definitions of agile software development and agility, in: European Conference on Software Process Improvement. Springer, pp. 247–258.

Laloux, F., 2014. Reinventing organizations: A guide to creating organizations inspired by the next stage in human consciousness. Nelson Parker.

Lapham, M.A., Miller, S., Adams, L., Brown, N., Hackemack, B., Hammons, C., Schenker, A., others, 2011. Agile methods: selected DoD management and acquisition concerns. Carnegie-Mellon Univ., Software Engineering Institute.

Lárusdóttir, M., Cajander, Å., Gulliksen, J., 2012. The Big Picture of UX is Missing in Scrum Projects., in: I-UxSED. pp. 42–48.

Lee, M.J., Ko, A.J., 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.

Liikkanen, L.A., 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, pp. 1095–1100.

Mao, J.-Y., Vredenburg, K., Smith, P.W., Carey, T., 2005. The State of User-centered Design Practice. Commun. ACM 48, 105–109.

Marshall, C., Rossman, G.B., 2014. Designing qualitative research. Sage publications.

Martin, A., Biddle, R., Noble, J., 2004. The XP customer role in practice: three studies, in: Agile Development Conference. Presented at the Agile Development Conference, pp. 42–54.

Matthiesen, S., Bjørn, P., Petersen, L.M., 2014. Figure out how to code with the hands of others: recognizing cultural blind spots in global software development, in: Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing. ACM, pp. 1107–1119.

McGinn, J., Chang, A.., 2013. RITE+Krug: A combination of usability test methods for Agile design. Journal of Usability Studies 8, 61–68.

McInerney, P., Maurer, F., 2005. UCD in Agile Projects: Dream Team or Odd Couple? interactions 12, 19–23.

McKay, J., Marshall, P., 2001. The dual imperatives of action research, in: Information Technology & People. pp. 46–59.

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, pp. 167–175.

Menéndez Blanco, M., Bordin, S., De Angeli, A., 2014. Socio-technical infrastructuring for participation.

Miller, L., 2005. Case Study of Customer Input For a Successful Product, in: Proceedings of Agile. pp. 225–234.

Moreno, A.M., Seffah, A., Capilla, R., Sanchez-Segura, M.-I., 2013. HCI practices for building usable software. Computer 100–102.

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.

Nielsen, J., 2006. Corporate Usability Maturity: Stages 1-4.

Nielsen, J., 1994. Usability engineering. Elsevier.

Nodder, C., Nielsen, J., 2010. Agile usability: Best practices for user experience on Agile development projects. Nielsen Norman Group.

Norman, D.A., 1986. Cognitive engineering. User centered system design.

Øvad, T., Bornoe, N., Larsen, L.B., Stage, J., 2015. Teaching software developers to perform UX tasks, in: Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction. ACM, pp. 397–406.

Øvad, T., Larsen, L.B., 2016a. How to reduce the UX bottleneck–train your software developers. Behaviour & Information Technology 35, 1080–1090.

Øvad, T., Larsen, L.B., 2016b. Templates: A Key to Success When Training Developers to Perform UX Tasks, in: Integrating User-Centred Design in Agile Development. Springer, pp. 77–96.

Øvad, T., Larsen, L.B., 2015. The prevalence of UX design in agile development processes in industry., in: Agile Conference (AGILE).

Patton, J., 2002. Hitting the target: adding interaction design to agile software development, in: OOPSLA 2002 Practitioners Reports. ACM.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J., 2008. The impact of agile practices on communication in software development. Empir Software Eng 13, 303–337.

Popli, R., Chauhan, N., 2012. Research challenges of agile estimation. Journal of Intelligent Computing and Applications.

Pries-Heje, L., Pries-Heje, J., 2011. Why Scrum works: A case study from an agile distributed project, in: Denmark and India. In Agile Conference (AGILE). IEEE. pp. 20–28.

Procter, R., Rouncefield, M., Poschen, M., Lin, Y., Voss, A., 2011. Agile project management: A case study of a virtual research environment development project. Computer Supported Cooperative Work (CSCW) 20, 197–225.

Provaglio, A., 2017. Shifting consciousness. Keynote talk at Agile Business Day, Venice.

Robey, D., Welke, R., Turk, D., 2001. Traditional, iterative, and component-based development: A social analysis of software development paradigms. Information Technology and Management 2, 53–70.

Robinson, H., Segal, J., Sharp, H., 2007. Ethnographically-informed empirical studies of software practice. Information and Software Technology 49, 540–551.

Rogers, Y., Sharp, H., Preece, J., 2011. Interaction design: beyond human-computer interaction. John Wiley & Sons.

Rönkkö, K., Dittrich, Y., Randall, D., 2005. When Plans do not Work Out: How Plans are Used in Software Development Projects. Comput Supported Coop Work 14, 433–468.

Runeson, P., Host, M., Rainer, A., Regnell, B., 2012. Case study research in software engineering: Guidelines and examples. John Wiley & Sons.

Salah, D., Paige, R.F., Cairns, P., 2014. A systematic literature review for agile development processes and user centred design integration, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, p. 5.

Sarma, A., 2005. A Survey of Collaborative Tools in Software Development (Technical report). Institute for Software Research, University of California, Irvine.

Schmidt, C., 2004. The analysis of semi-structured interviews. A companion to qualitative research 253–258.

Schmidt, K., 1994. Modes and mechanisms of interaction in cooperative work (Technical report). Risø National Laboratory, Roskilde, Denmark.

Schmidt, K., Bannon, L., 1992. Taking CSCW seriously. Computer Supported Cooperative Work (CSCW) 1, 7–40.

Schwaber, K., 2004. Agile project management with Scrum. Microsoft press.

Schwaber, K., 2000. Calculating Sprint Burndown and Velocity of Work.

Schwaber, K., Sutherland, J., 2011. The Scrum guide. Scrum.org.

Schwartz, L., 2014. Agile-User Experience Design: does the involvement of usability experts improve the software quality? International Journal on Advances in Software 7, 456–468.

Schwartz, L., 2013a. Agile-user experience design: with or without a usability expert in the team? Proceedings of the ICSEA 359–363.

Schwartz, L., 2013b. Agile-User Experience Design: an Agile and User-Centered Process?

Selic, B., 2009. Agile documentation, anyone? Software, IEEE 26, 11–12.

Sharp, H., Dittrich, Y., Souza, C.R.B. de, 2016. The Role of Ethnographic Studies in Empirical Software Engineering. IEEE Transactions on Software Engineering 42, 786–804.

Sharp, H., Robinson, H., Segal, J., 2008. Integrating user centered design and software engineering: a role for extreme programming. Retrieved from: http://www. ics. heacademy. ac. uk/events/presentations/376_hcie-arp2. pdf.

Silverman, D., 2013. Doing qualitative research: A practical handbook. SAGE Publications Limited.

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. IEEE, pp. V2–32.

StoryboardThat, URL http://www.storyboardthat.com/

Strauss, A., 1988. The articulation of project work: An organizational process, in: The Sociological Quarterly. pp. 163–178.

Strode, D.E., Huff, S.L., Hope, B., Link, S., 2012. Coordination in co-located agile software development projects. Journal of Systems and Software 85, 1222–1238.

Susman, G.I., 1983. Action research: a sociotechnical systems perspective. Beyond method: Strategies for social research 95–113.

Sy, D., 2007. Adapting usability investigations for Agile user-centered design. Journal of Usability Studies 2, 112–132.

Sy, D., Miller, L., 2008. Optimizing agile user-centred design, in: CHI'08 Extended Abstracts on Human Factors in Computing Systems. ACM, pp. 3897–3900.

The Standish Group's CHAOS Report, 2014. URL https://www.projectsmart.co.uk/white-papers/chaos-report.pdf

Ungar, J.M., White, J.., 2008. Agile user centered design: Enter the design studio – A case study, in: Proc. CHI 2008, ACM. Press, pp. 2167–2177.

Usman, M., Mendes, E., Weidt, F., Britto, R., 2014. Effort estimation in agile software development: a systematic literature review, in: Proceedings of the 10th International Conference on Predictive Models in Software Engineering. ACM, pp. 82–91.

Venturi, G., Troost, J., Jokela, Ti., 2006. People, organizations, and processes: An inquiry into the adoption of user-centered design in industry. International Journal of Human-Computer Interaction 21, 219–238.

Verdiesen, B., 2014. Agile user experience. MSc dissertation, Radboud University Nijmegen, Nijmegen.

Vessey, I., Sravanapudi, A.P., 1995. CASE tools as collaborative support technologies, in: Communications of the ACM. pp. 83–95.

Vygotskij, L.S., Cole, M., 1978. Mind in Society. Harvard university press.

Williams, L., Cockburn, A., 2003. Guest Editors' Introduction: Agile Software Development: It's about Feedback and Change. Computer 36, 39–43.

Wolkerstorfer, P., 2008. Probing an Agile usability process, in: Proc. CHI 2008, ACM. Press, pp. 2151–2157.

Wood, D., Bruner, J.S., Ross, G., 1976. The role of tutoring in problem solving. Journal of child psychology and psychiatry 17, 89–100.