

Interaction Design and Agile Development: A Real-World Perspective

by

Jennifer Ferreira

A thesis
submitted to Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Master of Science
in Computer Science.

Victoria University of Wellington
2007

Abstract

Although both agile development methods and interaction design aim to build quality software that meets the user's needs, each approaches development from a different perspective. Agile development methods describe activities addressing the coding (and in some cases the process management) part of software development, whereas interaction design methods describe activities for developing that aspect of the software that will be perceived by the user. Agile development and interaction design each have little to say about the other, despite the reality that both approaches are combined in practice. There has been little investigation into how the two processes work together, and the issues that arise. To aim for a better understanding of practice, we conducted grounded theory research about real-world software teams who combine interaction design and agile development. The results provided insights into interaction design being done up front (before implementation begins), the structure of the development iterations, the effect of incorporating interaction design techniques into the agile development iterations, as well as the role of the interaction designer on agile teams. The analysis also highlighted areas that may benefit from further research.

Acknowledgments

My thanks goes to:

- All the people around the world I've ever talked to about this research — this document owes its existence to your interest and input.
- The Agile Alliance for providing funds supporting this research.
- My supervisors, James Noble and Robert Biddle, for being so inspirational, encouraging, challenging and intelligent. Thanks also to Stuart Marshall who provided excellent support.
- My friends at Mt. Vic Dojo, the Wellington Gaelic Football players and the Vic UFO ultimate team for the welcome distraction.
- My family.

Contents

1	Introduction	1
1.1	Agile software development	3
1.2	Interaction design	4
1.3	Motivation for research	6
1.4	Research questions	7
1.5	Road map	7
2	Background	11
2.1	Software process research	11
2.2	Waterfall shortfalls	12
2.3	Agile as the answer	14
2.3.1	eXtreme Programming	16
2.3.2	Scrum	24
2.4	Interaction design	27
2.4.1	User-centered design and usability	27
2.4.2	Interaction design techniques	29
2.4.3	Interaction design and software development	31
2.5	Combining interaction design and agile development	32
2.5.1	BDUF considered harmful	32
2.5.2	A comparison	33
2.5.3	Related studies	36
2.6	Summary	37

3	Research Method	39
3.1	Why is grounded theory appropriate?	39
3.2	Grounded theory	41
3.2.1	Theoretical sampling	42
3.2.2	Theoretical saturation	42
3.2.3	Role of the literature	43
3.3	The researcher	43
3.4	The participants	44
3.5	Conducting the study	50
3.5.1	Participant recruitment	51
3.5.2	Interviews	51
3.5.3	Data analysis	53
3.5.4	Theory development	54
3.6	Reliability and validity of the study	54
4	Introduction to results	61
4.1	Stages of development	62
4.2	Interaction Design Approaches	62
4.2.1	Design strategies	64
4.2.2	Implementation strategies	65
4.3	Feedback and change	69
4.4	Interaction designers on agile teams	70
4.5	Publication of results	71
4.6	Next steps	71
5	Before implementation begins	73
5.1	Gaining a holistic view up front	74
5.2	Studying clients and users	75
5.3	Designing for change	78
5.4	Driving the development effort	79
5.5	Advantages of interaction design before development begins	81

5.5.1	Another notion of up front	84
5.6	Summary	85
6	Interaction design approaches	87
6.1	Overview of interaction design approaches	88
6.2	Comprehensive Design	89
6.2.1	Implementation as Refinement	93
6.2.2	Key conditions	96
6.3	Evolutionary Design	97
6.3.1	Implementation as Looking Ahead	102
6.3.2	Key conditions	107
6.4	Parallelisation	108
6.4.1	Key conditions	113
6.5	Summary	114
7	Inside iterations	117
7.1	Development iterations drive usability testing	118
7.2	Usability testing results in changes in development	121
7.3	Iterating with working software brings insights	124
7.4	Iteration planning affects interaction design	126
7.5	Experimentation and adjustment of process	128
7.6	Summary	134
8	Agility and the interaction designer	137
8.1	Usability has priority	137
8.2	Interaction design requires skill	140
8.2.1	When developers do interaction design	144
8.2.2	Interaction design skill contributes value	145
8.3	Interaction design as collaboration	146

8.3.1	Setting the target	147
8.3.2	Maintaining the target	149
8.4	Lurking pitfalls of the interaction designer role	151
8.5	Interaction designer: Shared vs. total control	153
8.6	Summary	159
9	Conclusion	161
9.1	Contributions	161
9.2	Returning to the literature	164
9.3	Topics for further discussion	166
9.3.1	Design: Agile values and cost of change	166
9.3.2	Implemented software vs. unimplemented prototype	170
9.4	Further work	174
A	Project profiles	177
A.1	P1	177
A.2	P2	179
A.3	P3	180
A.4	P4 and P5	182
B	Interview questionnaires	185
B.1	Initial	185
B.1.1	Background	185
B.1.2	Requirements	185
B.1.3	Teams	186
B.1.4	Process	186
B.1.5	Usability	187
B.1.6	Testing and refactoring	187
B.1.7	Users/customers	188
B.1.8	Wrap-up	188
B.2	Revised	189
B.2.1	Background	189

B.2.2	Experience with combining interaction design and agile development	189
B.2.3	Process	189
B.2.4	Teams	190
B.2.5	Wrap-up	191
C	Participant information and approved HEC application	193

Chapter 1

Introduction

The software industry is well aware that users prefer products that provide a satisfying, productive experience [16]. Satisfied users are loyal, translating to increased revenue for the developer company. CIO Insight reported that of 400 IT executives surveyed, 80% cited customer satisfaction as their top priority [17]. Despite the importance of user satisfaction, software products that meet the user's goals and provide a satisfying user experience are difficult to build — the literature and practice abounds with methodologies all aiming to take control of the chaos and complexity of software development to produce good software [20, 54, 59].

Agile development methodologies comprise an approach to developing software that meets the user's needs, by having the development team closely collaborate with a potential user, or customer, and regularly delivering working software to the customer. Interaction design presents another approach where potential users are central to the design of the user's interaction with a software product, to ensure a better fit between the end product and the user's needs. Unfortunately, agile developers and interaction designers approach software development from different perspectives. Agile development methods describe activities addressing the coding (or process management) part of software development, whereas interaction design methods describe activities for developing that aspect

of the software that will be perceived by the user. Agile development and interaction design each have little to say about the other, despite the reality that aspects of both approaches are combined in practice. Both agile development and interaction design have a major role in making good software, but there has been little investigation or discussion on how the two processes work together, and the issues that arise. This thesis reports on our grounded theory study of software teams that use both interaction design and agile development. Our aim was to better understand practice and as a result, our study uncovered insights into interaction design being done up front (before software development begins), the structure of the development iterations, the effect of incorporating interaction design techniques into the agile development iterations, as well as the role of the interaction designer on agile teams. The author of an article for *Interactions* magazine in 2004, speculated that the reason the agile community has focused little on user interaction design must mean “that either they neglect the user experience or are focusing on projects with less need for sophistication in user experience” [4]. The results of this thesis suggest this to be a false assertion. Evidence from agile practitioners in this and other studies, suggests the user experience is regarded very highly by the agile community and the user experience of the products developed using agile methods are indeed highly sophisticated. Yet, the lack of guidance for software development teams combining interaction design and agile development, means that teams are left to innovate their own approaches. We report on these approaches in this thesis, but also highlight the issues that may improve understanding with further study. Section 1.1 introduces agile development and section 1.2 introduces interaction design. Section 1.3 describes the motivation for this research, section 1.4 presents the specific research questions for this study and section 1.5 explains the structure for the following chapters of this thesis.

1.1 Agile software development

Agile, a term adopted by practitioners in early 2001 [54], is generally used to refer to the collection of iterative, lightweight development methods that share a set of values and principles articulated by the agile manifesto [55]. The values are as follows:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

These agile values have been formulated to counter the more plan-driven, overhead rich development methods commonly described as *waterfall* or *traditional* [115]. In waterfall software development processes, development activities are performed in sequence, the next activity only commencing once the previous one has completed. Further, the product and the process is expected to be heavily documented. Developing software in this way constrains change, in that new things that are learned by the development team or the client is incorporated into the product with much difficulty [54]. Increasingly, practitioners have reported that waterfall software development methods are inadequate for dealing with the environment of change in which much software development occurs today [71, 103, 147]: Besides often being delivered late, these software products were found to have quality and usability issues [109].

Agile methods were developed to cope with the inevitable changes that occur over the course of a software development project — requirements change, technology changes, people are added and taken off the team, etc. — while still delivering a high quality product [147]. Agile is not a methodology in itself — there are several distinct methodologies that Fowler refers to as ‘philosophies’ of software development [54] with different approaches, such as eXtreme Programming (XP) [9, 10], Scrum [124],

Crystal Clear [31] and Feature Driven Development (FDD) [107]. Agile methods emphasise developing a software product incrementally, gaining feedback from stakeholders after each increment and iteratively incorporating that feedback back into the subsequent development. Cockburn notes, “There is no substitute for rapid feedback, both on the product and on the development process itself” [29]. An important difference in agile development, compared to the waterfall model, is that the software development activities are repeated with each increment and each iteration deals with only the set of requirements that are known at that time — allowing feedback to influence the product early on in the development process and for emergent requirements to be incorporated.

Due to their success, agile methods are fast becoming mainstream in the software industry. Sections 2.3.1 and 2.3.2 introduce XP and Scrum — the two agile development methods used by the participants in our study.

At this point, it is necessary to point out the aversion of agilists to the term *process* and what it implies. In accordance with the pragmatic aspects of the values of the agile manifesto given above, Highsmith explains: “Process deals with prescription and formality, whereas practice deals with all the variations and disorderliness of getting work done” [70]. In this thesis, when the term *process* appears, it is not intended to refer to the ‘prescription’ and ‘formal’ aspects of any method, but is rather intended to convey the collection of software development activities that are realised during the course of the software development project.

1.2 Interaction design

From Verplank’s use of the term *interaction design* [144], it has since been widely adopted by practitioners and researchers [36, 126]. Cooper defines interaction design as “[referring] to the selection of [software] behavior, function and information and their presentation to users” [36, p22]. In this thesis, we broaden Cooper’s definition, so that when we refer to in-

interaction design, we intend to include user research, user modeling and evaluation of the design, to better distinguish between agile development activities and activities targeting the user experience. The activities of interaction design result in the user interface (UI) of the software product — “those points of contact between systems and their users” [34, p5]. The Human-Computer Interaction (HCI) literature is dominated by the user-centered design (UCD) approach to interaction design, which advocates that the end users are an integral part of the design process and should influence the resulting design [104, 105]. Being user-centered implies a software product meets the goals of its end users and adheres to the ISO 9241-11 definition of *usability*:

The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

HCI researchers agree that the best way of achieving a product with a good level of usability is by obtaining accurate and frequent user input [43, 57, 106, 126]. UCD practitioners obtain user input by means of user evaluations of designs and improving the design based on user feedback, thereby refining the design in an iterative fashion.

Importantly, the HCI community have accepted that UCD activities, such as user evaluations and iterative UI design, occur before the product is implemented. Then the interaction designers hand off the design to the developers who are responsible for implementing the product. Consequently, the UCD approach to software development, similar to the waterfall approach, depends heavily on the interaction designers having access to the full set of requirements for a product at the outset of the development effort.

1.3 Motivation for research

The agile manifesto prefers “Responding to change over following a plan,” whereas the user-centered approach to interaction design advocates an intensive up-front user research period, followed by iterative evaluations of designs with users — often referred to as Big Design Up Front (BDUF) [6]. The apparent contradiction between agile values and the BDUF approach has sparked much debate regarding the place of interaction design within agile development, notably between Kent Beck, creator of eXtreme Programming (XP) and Alan Cooper, creator of Goal-Directed Design [36]. While there was much agreement, the main point of difference was that Cooper advocated UI design being done entirely before any development, using an iterative approach with lightweight prototypes, as working software would be too expensive and time-consuming. Beck argued that the two iterative processes might work together and that use of XP practices meant that working software would be cost and time effective in comparison with prototypes [97]:

Cooper: “I’m talking about incorporating a new constituency that focuses exclusively on the behavioral issues. And the behavioral issues need to be addressed before construction begins. ”

Beck: “The interaction designer becomes a bottleneck, because all the decision-making comes to this one central point. The process, however, seems to be avoiding a problem that we’ve worked very hard to eliminate. The engineering practices of extreme programming are precisely there to eliminate that imbalance ...”

Agile methods and interaction design do not give any guidance regarding each other and there has been little investigation or discussion about the issues that arise from their combination. Both have a major role in making good software, despite their different perspectives on creating software. Interaction design focuses on how the end users will work with the software and agile development focuses on how the software should be con-

structured, or how the development process should be managed.

The way these two perspectives are being combined in practice is still unclear. Studies dealing with combining interaction design and agile software development have been largely anecdotal, and consequently, real-world agile teams are left to devise their own strategies for combining agile methods with interaction design.

Only recently, relevant research into practice has emerged (see section 2.5) and this thesis hopes to contribute to the existing research by drawing on the experience of a diverse group of practitioners in the field.

1.4 Research questions

The aim of this study is to answer the very broad question “How do real-world agile teams combine interaction design with their agile development activities?” We break this question down into:

- How do agile teams sequence their development and interaction design activities?
- What processes and techniques do agile teams follow when combining interaction design activities with agile development activities?
- How does agile development affect interaction design when combined, and vice versa?
- What are the effects of agile development on the interaction designer role?

1.5 Road map

Chapter 2 highlights the characteristics of the software development environment, which agile development attempts to address, describes eXtreme

Programming and Scrum and compares interaction design and agile development.

Chapter 3 discusses the research method followed for this study with a brief introduction to grounded theory, including the appropriateness of using the grounded theory approach, details about the researcher and research participants and the validity and reliability of the results.

Chapter 4 describes the large-scale features that characterised our participants' approaches to combining interaction design activities with agile development activities. The introduction given here provides the context for the discussion throughout the rest of the thesis, with the substantiating quotes from the participants presented in the following chapters.

Chapter 5 discusses the findings as they relate to the software development activities that took place before the implementation stage of the development effort began, i.e., up-front.

Chapter 6 focuses on the way the teams organised their interaction design and agile development activities and their resulting approaches to interaction design, in terms of the design and implementation.

Chapter 7 shows how agile development iterations were found to be opportunities for obtaining feedback and incorporating change in the interaction design and presents some of the areas where agile development enabled the activities of interaction design and vice versa.

Chapter 8 discusses the role of the interaction designer, within the context of the agile development environment, as it emerged from the interview data.

Chapter 9 concludes the thesis with the major findings and future work.

Chapter 2

Background

This chapter highlights the characteristics of the software development environment, which agile development attempts to address. eXtreme Programming and Scrum and interaction design are described, and then interaction design and agile development compared. The next chapter discusses the research method followed for this study.

2.1 Software process research

According to Fuggetta, a *software process* can be defined as “the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product” and research in this area assumes that the quality of the process influences the quality of the resulting product [56].

Developing software is complex [20, 19] and the challenge still remains to create and organise the development activities in such a way that quality software is delivered on time and within budget. Over time, numerous approaches have emerged to manage the software process so that a high quality product can be achieved more efficiently. Li et al. refer to the collection of these approaches as the “Software Process Spectrum” [85]. At one end there are lightweight agile methodologies and on the other, heavy-

weight waterfall processes. Light- or heavyweight in this sense referring to the amount of deliverables, or artifacts, to produce for the benefit of the management of the software development process — all in an attempt to instill order on the perceived chaos [33].

2.2 Waterfall shortfalls

Software development processes possessing either or both of two key characteristics can be considered *traditional* [115]. The first relates to the direction of development: a linear progression from inception to delivery, resulting in a process that is governed by a plan formulated at the initial stages of development and rigidly adhered to for the duration of the project. The second refers to the amount of overhead produced during the development effort, in the form of documentation or deliverables.

The most well-known of the traditional methods is the *waterfall* method — attributed to Dr. Winston W. Royce, despite the fact that he never referred to the process he described as waterfall and his paper does not support the idea of the development process as a strict progression from one step to the next. The waterfall label has remained however, due to the widely adopted version of Royce's model where feedback into previous stages is discouraged and development continues in a linear fashion to completion, as water flows over a waterfall.

In terms of producing deliverables, Royce's model is certainly heavyweight:

How much documentation? My own view is 'quite a lot;' certainly more than most programmers, analysts, or program designers are willing to do if left to their own devices. The first rule of managing software development is ruthless enforcement of documentation requirements ... Management of software is simply impossible without a very high degree of documentation [118].

Royce proposed seven stages of software development [118]:

1. System requirements
2. Software requirements
3. Analysis
4. Program design
5. Coding
6. Testing
7. Operations

The software requirements stage delivers the first document, known as the 'Software Requirements' document, the program design stage delivers the 'Test Plan', the 'Interface Design Specification' and the 'Final Design Specification,' and the final stage the 'Operating Instructions' for the system [118].

While the sequential waterfall approach has afforded project managers a sense of control [76, 143] and exacted a kind of discipline from those involved in the development project [118], it has not proven itself a resounding success in the software development arena. Considering the published figures touting the poor performance of traditional projects in the mid 1990s, it is not surprising that practitioners were turning their attentions to alternative approaches [55]:

- Rodrigues and Bowers reported on the increasing trend of budget over-runs of 40–200% [117].
- The Standish Group reported that only 16.2% of software projects in the United States were completed on time and within their budget [141].

- According to the Special Interest Group concerned with the Organisational Aspects of IT (OASIG), in 1996, just 10-20% of software products deployed in the United Kingdom met all their success criteria [26].

In 1988 Boehm had already recognised the weakness of the waterfall development process as forcing the completion and documentation of poorly understood requirements and design, leading to “large quantities of unusable code” [15]. This was found to be especially true in the case of end-user interactive applications.

2.3 Agile as the answer

Change, speed and uncertainty have proven problematic for projects following the waterfall approach [125]. While Royce advocated restraining changes from the customer with early commitments [118], agile practitioners have responded with more adaptive approaches:

Agility, for a software development organisation, is the ability to adapt and react expeditiously and appropriately to changes in its environment and to demands imposed by this environment. An agile process is one that readily embraces and supports this degree of adaptability. So, it is not simply about the size of the process, or the speed of delivery; it is mainly about flexibility [78].

Agile development, unlike the waterfall approach, adapts to change rather than adhering to a plan conjured up at the outset of the development project. The mechanism which allows this adaptability is *iterative development*, an established practice of software development since the 1950s [80]. Iterative development provides agile teams with regular intervals for obtaining feedback about the process and the product. Based on this feedback, the team can determine whether the process and the product are still applicable to the current situation and adjust as necessary [54]. A

typical agile development iteration is illustrated in figure 2.1. At the beginning of the iteration a set of requirements, or features, is selected and prioritised with the customer, after which the developers set about implementing those features. The implementation work only lasts for the length of one iteration — from one week to one month — at the end of which the implemented product is evaluated, along with the process. The team assess the accuracy of the work estimates created at the iteration planning and ask questions such as: “What did the team do well?”; “What can be improved?”; etc. [42]. If, at the end of the iteration, there are still outstanding features to be implemented, then the next iteration is planned and carried out and repeated until the customer agrees that the required features have been implemented.

The activity sequence of plan–implement–evaluate is apparent in different flavours of agile, for example, eXtreme Programming (XP) [9, 10], Scrum [124], Crystal Clear [31] and Feature Driven Development (FDD) [107].

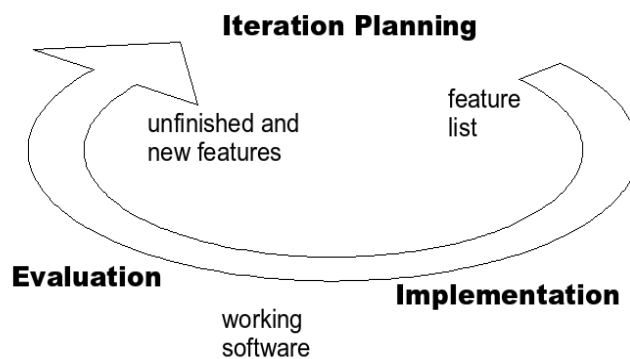


Figure 2.1: Agile Iteration

In agile development the emphasis on documentation is low, in contrast to the traditional approach, favouring continual involvement of a

customer¹ throughout the development effort — hence, the association of the term ‘lightweight’ with agile methods.

Figures on agile project success and adoption are becoming more prevalent in the literature. For example, Larsen, a board member of the Agile Alliance², provides evidence that agile methods are becoming more well-known in industry [81]. Similarly, Ambler, an experienced agile practitioner and author, conducted a survey of agile in practice. With 4,232 respondents, the results indicate that agile methods are gaining acceptance within the IT industry [3]. The Standish Group produced another report in 2006 that stated 41% of agile projects succeeded, compared to 16% of traditional projects [119]. The Standish Report, along with the numerous experience reports in the literature reporting successful agile adoption, suggests that agile software development can be a viable alternative to the traditional approaches. In sections 2.3.1 and 2.3.2 we describe the two agile development methods that were followed by the participants in our study — eXtreme Programming and Scrum — and present published reports of their success.

2.3.1 eXtreme Programming

eXtreme Programming (XP) is regarded as the most popular agile method in use today [33, 54]. In 1999 Kent Beck’s influential book: *Extreme Programming Explained: Embrace Change* introduced the software development world to this agile method with its coherent set of values, principles and practices. Revised in 2004, in a second edition, Beck states that the goal of XP is “outstanding software development in an environment of vague or rapidly changing requirements” [10].

The driving values of XP, which underlie the principles and practices are:

¹The customer may be a potential end user of the product under development or a business representative from the client company.

²<http://www.agilealliance.org>

- **Communication** In order for the team to cooperate effectively, good communication is essential. Learning from other team members can help to avoid mistakes from the past.
- **Simplicity** Beck asks the question “What is the simplest thing that could possibly work?” to encourage teams to remove unnecessary complexity and to concentrate on producing working software that adhere to today’s requirements, as opposed to implementing features that may or may not be required in the future.
- **Feedback** Not understanding the requirements at the outset of the project, the team needs to generate feedback cycles that are as short as possible in order to improve the software and get closer to reaching their goals with that software.
- **Courage** There are several ways in which the value of courage can be fostered on an XP team. Team members are encouraged to communicate with other team members when they see a better way of performing some task. Having the courage to speak up in such situations is seen as valuable.

Beck added a fifth value in the second edition of his book [10]:

- **Respect** The team members need to respect each other as well as the project they are working on in order for XP to work. Equality among team members is particularly vital and no one is seen as more important.

Each development project is unique and the environment in which it occurs is also unique to every project. For this reason, Beck presents 14 principles of XP, grounded in the driving values, for the cases where the team’s environment is not suitable for adopting a prescribed practice. Teams can innovate development practices based on the guiding principles: (1) humanity, (2) economics, (3) mutual benefit, (4) self-similarity, (5) improvement, (6) diversity, (7) reflection, (8) flow, (9) opportunity, (10) redundancy,

(11) failure, (12) quality, (13) baby steps and (14) accepted responsibility. We will not investigate the principles further here, but will instead focus on the practices of XP.

XP practices

The practices of XP are considered the embodiment of the driving values [116], that teams can apply in order to get their work done [9, 70]. The practices are not new to software development — several practices have been in existence before the XP methodology had been formulated, but their effectiveness has been proven in environments of rapidly changing requirements [8]. Notably, Gilb’s evolutionary delivery of software [60], which he describes as “actually delivering final results, final requirements to real users and stakeholders” [32], is central to XP.

Beck’s first edition gives the following twelve practices that structure the development activities [9]:

1. **The Planning Game** Prioritise the required features of the system and, based on the work estimates, select the work for the next increment.
2. **Small releases** Iteratively add to the features in the system in short cycles.
3. **Metaphor** Base development and communication about the development work on a shared ‘story’ or metaphor.
4. **Simple design** Create the simplest system that will work, removing complexity as soon as it is discovered.
5. **Testing** All code has to pass automated unit tests before becoming part of the overall system and the customer accepts the system when it passes all functional, or acceptance, tests.
6. **Refactoring** Improve the design of the current system.

7. **Pair programming** Two people write code together at a single machine, collaborating on all aspects of the programming effort.
8. **Collective ownership** The whole team can make improvements and changes to the code base.
9. **Continuous integration** Every few hours changes should be integrated into the larger system. This helps minimise the time required to integrate changes, as the changes are likely to be smaller.
10. **40-hour work week** Limit work hours so as not to work overtime.
11. **On-site customer** Ideally an end user, the customer is in the same location as the development team, able to answer queries as they arise.
12. **Coding standards** A consistent standard that the whole team can adhere to when coding.

The following practices are taken from the second edition of Beck's book, which are not only considered a refinement of the practices in the first edition, but also practices that development teams new to XP can start with [10]:

1. **Sit together** When working, the team sits within eye contact of each other, surrounded by artifacts displaying project status. Usually this implies the team is small enough in order for every person to fit in the same room.
2. **Whole team** Every individual's contribution, in terms of knowledge, skills and perspective, influences the success of the project. Therefore, the sense of "team" is required for trusting, collaborative work to occur.
3. **Informative workspace** The workspace should make visible the status of the project, support communication within the team about the

project, provide a space where people can be comfortable and have positive social interactions.

4. **Energized work** The team should only be working the number of hours they can productively focus on their work.
5. **Pair programming** Similar to the practice as explained in the first edition.
6. **Stories**(Also known as User Stories) are created by the development team and the customer to illustrate how a user might use the system under development. Kent Beck requires that user stories are
 - **Testable** —You can write automatic tests to detect the presence of the story.
 - **Progress** —The customer’s side of the team is willing to accept the story as a sign of progress toward their larger goal.
 - **Bite-sized** —The story should be completable within the iteration.
 - **Estimable** —The technical side of the team must be able to guess how much of the team’s time the story will require to get working.
7. **Weekly cycle** Work should be planned only for the next week.
8. **Quarterly cycle** Quarterly reflection about the team, the project and its progress aligns well with other business activities that occur quarterly, however, any natural timescale that the team can agree on will suffice. The intention is to allow a large enough interval for progress to occur and still have regular opportunities to evaluate the situation.
9. **Slack** Including minor tasks in the development plan that have little impact on the system if they are not implemented, provides a buffer

to the development and increases the team's chances of completing the more important tasks on time.

10. **Ten-minute build** Building the system and running all of the tests should take ten minutes at the most.
11. **Continuous integration** Similar to the practice as explained in the first edition.
12. **Test-first programming** Before writing any code, developers should create an automated test. Testing is an effective way of determining concisely what needs to be implemented, writing tests signal characteristics of the system design — tests should be easy to write, builds trust among the developers that all the code produced passes the required tests and helps developers to focus on completing a specific task instead of wasting time playing with possible solutions.
13. **Incremental design** System design is not discouraged in XP per se, it is simply not considered useful when it is completed before implementation begins. Instead, making constant improvements to the design as the system is implemented incrementally, helps keep the cost of change low. Improving system design via refactoring is a well-known technique employed by XP teams.

Clearly, the practices are not only intended to structure software development activities, but also to structure the environment in which these activities take place. Cockburn's description of XP in action, from the developers' perspective, presents a context in which XP teams do their work:

It calls for all the developers to sit in one large room, for there to be a usage expert or 'customer' on the development staff full time, for the programmers to work in pairs and develop extensive unit tests for their code that can be run automatically at any time, for those tests always to run at 100% of all code that is checked in, and for code to be developed in nano-increments, checked in and integrated several

times a day. The result is delivered to real users every two to four weeks.

In exchange for all this rigor in the development process, the team is excused from producing any extraneous documentation. The requirements live as an outline on collections of index cards, and the running project plan is on the whiteboard. The design lives in the oral tradition among the programmers, in the unit tests, and in the oft-tidied-up code itself. [31, p29]

Key roles

The XP team are considered stakeholders in the development project, i.e., they share equal responsibility and accountability for the project's outcome. The roles are also not rigid and any team member can contribute in the best way they can albeit in several different roles [9, 10]. Therefore, there may be more than one individual in each role and one individual may have more than one role on the team. Beck identifies four key roles: customer, programmer, coach and tracker:

- **Customer** Understands the domain for which the software is being developed, understands what generates business value for the client company, can recognise when the software system delivers that value and can prioritise the requirements [11]. Further, the customer is ideally on-site and accessible to the rest of the development team throughout the development effort.
- **Programmer** Or Developer, has the skills to translate the requirements into code that passes all tests and adheres to a simple design.
- **Coach** Understands XP and software development and can guide and mentor the team.
- **Tracker** Keeps track of the schedule and the team's progress.

In the second edition of Beck's book, there is acknowledgment of an **Interaction Designer** role, which is expected to work closely with the customer role, however, there is little guidance on how XP supports their work or how interaction design informs XP — other than the conjecture that "Interaction designers specify a little bit up front and continue to refine the user interface throughout the life of the project" [10].

XP success stories

XP was first used on the Chrysler Comprehensive Compensation (C3) project to rewrite the Daimler-Chrysler payroll package [140, 148]. Since its initial success, XP has been successfully applied to numerous software development projects. There has been reported success of XP adoption by development teams ranging in size — from small teams [82], through medium sized teams [72], to large [75, 88] and distributed teams [18, 67]. Software projects in different application domains (for example web-based application development [67], maintenance environments [111] and financial services [103]) and at different stages of completion (greenfield development [114], existing project extension [82]) have employed XP and achieved their success criteria. The current consensus appears to be that XP can be applied to a wide range of software development projects in various environments of vague and changing requirements, as long as there is buy-in from all people involved in the development effort. Although XP was initially developed for small to medium-sized projects, the numerous successful adoption experiences in the industry has prompted software companies to extend the application of XP to large scale development projects and there is evidence in the literature that in order to do this XP is often adjusted. Practices may be added or changed to suit the needs of a large development team [79], which agrees with Beck's vision of the future of XP [7]:

[..] it is and should always be an evolving fabric of mutually

supporting practices. These practices will continue to evolve, perhaps radically, over the next few years.

2.3.2 Scrum

Named after the *scrum* in the sport of rugby, which is “a tight formation of forwards who bind together in specific positions when a scrum-down is called,” [58] Scrum follows XP in popularity [3]. First observed by Takeuchi and Nonaka [138], Scrum is fully described in Schwaber and Beedle’s book published in 2001 [124]. Instead of defining how software should be built, Scrum is an empirical approach that accepts software development as a non-linear, ill-defined activity requiring frequent inspection and adaptation. Schwaber describes Scrum as “operating at the edge of chaos,” which aims to “operate adaptively within a complex environment using imprecise processes” [122]. Due to the unpredictable nature of software development, Scrum consists of activities that manage tasks such as backlog, work, risk, issues, problems and changes so that after each iteration the delivered solution, as a product of time, cost, functionality and quality, is the best one possible [123]. The implementation activities concerned with building the product are decided on by each individual team for each project.

The structure of Scrum

Scrum consists of three main phases, of which only the first and last are defined processes [122]:

1. **Pregame** The Pregame phase is defined in terms of planning and system design activities. In the same way that a list of user stories are created in XP, a list of backlog items are created in Scrum, which define the features of the product that will be iteratively implemented during the Game phase. Estimation of releases and delivery of the final product are also created, along with other administra-

tive project management issues, such as the verification of management approval and funding and risk management. A certain level of domain analysis is performed in order to ensure domain models are current and up to date.

2. **Game** The Game phase is concerned with implementation activities and is further broken up into iterative development units called *sprints*. Sprints usually last about one month, during which daily Scrum Meetings are held to determine [12]:

- What items were completed since the last Scrum Meeting.
- What issues or blocks have been found that need to be resolved.
- What new assignments make sense for the team to complete until the next Scrum Meeting.

At the end of the sprint, the product is demonstrated to the customer during a *Demo* session, after which the backlog items are reorganised and a new sprint is started. Scrum has no prescribed implementation practices for the Game phase, and teams have been known to apply the XP implementation strategies [1, 77].

3. **Postgame** The Postgame phase, or Closure, occurs when the product is considered ready to be released to the client. Closure tasks include integration testing and finalising documentation.

Key roles

Sutherland summarises the three key roles in the Scrum team as follows [134]:

- **Product Owner** Defines the features of the product, decides on release date and content, is responsible for the profitability of the product (ROI), prioritises features according to market value, can change

features and priority every 30 days and accepts or rejects work results.

- **Scrum Master** Ensures that the team is fully functional and productive, enables close cooperation across all roles and functions and removes barriers, shields the team from external interferences and ensures that the process is followed. Invites to daily scrum, iteration review and planning meetings.
- **Team** Cross-functional, seven plus/minus two members, selects the iteration goal and specifies work results, has the right to do everything within the boundaries of the project guidelines to reach the iteration goal, organises itself and its work and demos work results to the Product Owner.

Scrum success stories

Despite the numerous reports of cervical spine injuries inflicted by rugby scrums, the Scrum development approach has enjoyed success in high profile companies such as Fuji-Xerox, Canon, Honda, NEC and Hewlett-Packard, as reported by Takeuchi and Nonaka [138]. Sutherland reported on introducing Scrum to projects of varying sizes and concluded that Scrum can apply in any development environment [133]. Some of the most recent success stories in the literature are: Sutherland, Jakobsen and Johnson reporting on Scrum success on large defense and health care projects [135], Cloke reporting on successful Scrum adoption at Yahoo! Music [27] and Smits and Pshigoda reporting on a company that provides IT Infrastructure management support and its transition to Scrum in a distributed team environment [131]. In all these experience reports, Scrum is clearly seen as a means to increased productivity, lower defects and scalability on software projects.

2.4 Interaction design

Defining *design* is highly dependent on context and varied [139, 152]. In HCI research the word is often preceded by other terms such as *user interface*, or *user experience* [57]. Even in the context of HCI, understanding of these concepts is not homogenous and with each term comes an attempt to describe a different perspective on designing a software system for users. Winograd gives an in-depth explanation of interaction design [149] but for the purposes of this thesis we will use the definition based on Cooper, already stated in section 1.2: “the selection of [software] behavior, function and information and their presentation to users,” [36, p22] including user research, user modeling and evaluation of the design.

2.4.1 User-centered design and usability

Models of interaction assume that the user has certain goals that they wish to achieve with a software system [43, p104]. Interaction design is directed toward designing a user interface that successfully translates between the software system and the user. Therefore, when designing interaction, designers take into account users’ needs, which tend to change with differing experience levels, cognitive and physical abilities, work environments, personalities, culture and age [127].

The term ‘user-centered design’ became widely used after the publication of Norman and Draper’s book entitled: *User-Centered System Design: New Perspectives on Human-Computer Interaction* [106]. In practice, the emphasis on users implies that creators of a software product need to research the users of their product and their needs, model information about the users and then iteratively evaluate the product with those users. The resulting *usability* of the product will depend on how easy it is to learn to use, how efficiently users can complete their tasks, the ease with which users can remember how to use the product, the amount of errors they make in using the product and how satisfied they are with the overall ex-

perience [102].

Experts have suggested measurable qualities of interaction design that determine how usable a software product is. These qualities help to specify an objective measure of usability that can be determined during usability testing. Shneiderman and Plaisant propose [127, p16]:

- Time to learn
- Speed of performance
- Rate of errors
- Retention over time
- Subjective satisfaction

Constantine and Lockwood present a similar list [34, p7]:

- Learnability
- Rememberability
- Efficiency in use
- Reliability in use
- User satisfaction

Based on the above qualities, usability of an interaction design can be seen to depend on whether the user can perform their tasks with the system and whether they have a pleasant experience in doing so. If any of the qualities listed above are not met, for example, if it is difficult to remember how to carry out a task when coming back to the system after a long absence, the usability is adversely affected.

Investing in usability requires scarce resources and software development companies consider investing to be worthwhile if the effect on their

Return on Investment (ROI) is positive. Studies have shown that investment in the usability of a product can lead to significant positive effects on ROI: Withrow et al. compared usability studies of a redesigned state government web portal to a previous version, and estimated that the new version generates an extra \$552,000 per year [150]; Cisco improved the navigation in the user interface of their intranet system and estimated that more productive users saved the company \$3 million and the standards established during this effort have cut development time and saved \$45.7 million [83]; Wixon and Jones report that Digital Equipment Corporation (DEC) estimated an 80% increase in revenue to be attributable to usability improvements [151]; The Nielsen Norman Group have performed research into the effect of ROI on usability investments for web sites and concluded that a 10% investment results in a 135% improvement of the desired metrics [101].

2.4.2 Interaction design techniques

User-centered design suggests that techniques for researching the users of the product and their needs, modeling information about the users, as well as iteratively evaluating the product with those users, are central to the design process [106]. Organisations and researchers have established standards and guidelines to which interaction designers may refer when real users are not available. In practice, if a development effort includes no explicit techniques dealing with the interaction design of the product under development, the interaction design emerges as a by-product of development, and results in a product with poor usability [47, 112].

User research

As part of capturing the requirements for a software project, interaction designers may obtain data from surveys, interviews, or in-situ observation sessions [126].

Modeling users

Using the data gathered during the user research activities, the interaction designer may model the information about the users using personas (user archetypes) [36] or scenarios (narrative descriptions of user tasks within a context) [22].

User evaluation

Once an interaction design has been constructed, regardless of the fidelity, interaction designers may have users evaluate the usability of their designs in several ways. One way may be to have the user walk through a design, known as a *walkthrough* or *user review*, where the design may be in the form of a prototype [34]. Another evaluation technique is known as storyboarding [43]. As in the film industry, storyboards provide snapshots of sequences of interaction.

For evaluating an implemented interaction design a laboratory evaluation session can be used to control interaction with the implementation for collecting statistical data about the structure, semantics and procedures within the user interface [34].

Design rules

Users are not always accessible during software development and interaction designers may have to rely on other sources of information on which to base their design decisions. One source is design rules that are supported by psychological, cognitive, ergonomic, sociological, economical or computation theory [43]. Design rules may be national or international standards as set by organisations for interaction designers to comply with. Design rules may also be guidelines regarding data entry, data display, sequence control, user guidance, data transmission and data protection [130]. Jakob Nielsen has famously collected and distilled ten design rules, or *heuristics*, to which interaction designers can refer during their design

process [98]. For example, the heuristic ‘Visibility of system status’ implies that the system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

An evaluation method based on heuristics, is known as *heuristic evaluation* [100] — a usability inspection method in which usability experts inspect the user interface and evaluate how well it adheres to a set of heuristics. A study by Nielsen has shown that a team of four or five evaluators, along with Nielsen’s ten usability heuristics, can identify approximately 75% of a software system’s usability issues [98]. Further, a study by Nielsen has shown heuristic evaluation to have a benefit-cost ratio of 48:1 [99].

2.4.3 Interaction design and software development

Like Norman [105], influential interaction design gurus Cooper [35] and Constantine and Lockwood [34] advocate interaction design activities such as user research, user modeling and prototyping, as occurring ‘up front’ on a software development project, i.e., before the coding activities begin. Cooper sees interaction design as part of the planning aspect of software development [97]:

I’m advocating interaction design, which is much more akin to requirements planning than it is to interface design [...] the behavioral issues need to be addressed before construction begins.

This is also the generally accepted view of the place of interaction design in practice. The assumption within this view is that all requirements for the system are known up front:

For user interfaces, the architecture — the overall organization, the navigation and the look-and-feel — must be designed to fit the full panoply of tasks to be covered [33].

As a result, the interaction designers complete the user research, modeling and prototyping in a user-centered way, up front, so that the complete

design can be implemented by the developers. In this way interaction design becomes a plan-driven approach to software development.

2.5 Combining interaction design and agile development

The way in which interaction design and agile development should work together has been discussed surprisingly little. One important early exception was the debate between Kent Beck and Alan Cooper [97]. This debate explicitly addressed the issue of when interaction design should occur relative to software development. In the debate Cooper argues that all the interaction design should be done before any coding, but Beck strongly disagrees, saying that the interaction designer becomes a bottleneck for the development team (see section 1.3).

2.5.1 BDUF considered harmful

The Portland Pattern Repository³ describes BDUF: “The term Big Design Up Front is commonly used to describe methods of software development where a ‘big’ design is created before coding and testing takes place.” Fixing the requirements up front in the form of a design belongs to the realm of predictive methods [52, 53] — precisely the approach to software development that agile methods were intended to counter. The agile manifesto includes the preferences: “Working software over comprehensive documentation” and “Responding to change over following a plan,” which are intended to discourage fixing requirements with plans and documentation, since it has been shown that the more the design is determined up front, the more expensive it is to change in the future [14]. Further, when change is inevitable, the time and effort put into creating the big design up front is seen as wasted [9, 87].

³<http://c2.com>

2.5. COMBINING INTERACTION DESIGN AND AGILE DEVELOPMENT33

	Interaction design	Agile development
Similarities	1. Focus on the user 2. Iterative	1. Focus on the customer 2. Iterative
Differences	1. Holistic 2. Specialised skills	1. Incremental 2. Team of generalists

Table 2.1: Comparing interaction design and agile development

There is a clear contradiction in approach between interaction designers who mirror Cooper’s sentiment and agile proponents who consider up-front interaction design as BDUF, leaving the topic of how interaction design and agile development should be combined wide open.

2.5.2 A comparison

To aim for a better understanding of how interaction design and agile development might work together, we propose some observations gleaned from the literature about their superficial similarities and differences — summarised in Table 2.1.

The agile value of “Customer collaboration over contract negotiation” mentioned above, along with two of the twelve agile principles [55],

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

...and ...

Business people and developers must work together daily throughout the project,

emphasises close collaboration with the customer and business people throughout the agile development effort. Interaction design’s focus on the user, through user-centered design, shows great potential for successfully combining interaction design and agile development [93]. Cooper suggests that the interaction designer acts a bridge between the customer and the developers, or programmers [97]:

During the design phase, the interaction designer works closely with the customers. During the detailed design phase, the interaction designer works closely with the programmers.

A study by Härtinen suggests that the interaction designer should be in the XP customer role [69] and Martin, Biddle and Noble found that this could be the case in practice [91].

A closer examination of the relation between the customer and agile development, and the user and interaction design, reveals a difference in perspective from which the user and customer is approached. Whereas agile development involves the customer to ensure all the required functionality has been implemented, interaction design adds the dimension of user satisfaction, i.e., does the user have a satisfying experience interacting with the product.

The iterative approach to development is another obvious similarity. Iteration has always been a characteristic of user-centered design and a central feature of the codification of approach in ISO standard 13407 *Human-centered design processes for interactive systems* [45, 73]. As discussed in section 2.3, iterative agile development implies that the software is developed starting with a set of requirements that are implemented and evaluated within a relatively short period of time, before adding the next set of requirements. Iterative interaction design, as discussed in section 2.4, takes place as a design, perhaps in the form of a prototype, is evaluated with a user and then redesigned and re-evaluated until consensus is reached that the design meets the user's needs.

On the surface, the iterative approaches of both agile development and interaction design seem to be similar, but as with the seemingly similar focus on the user and customer, the iterative approach of agile development and interaction design masks an underlying difference. Interaction design and agile development differ in the artifact that each iterates on. Interaction design iterates on a *representation* (usually a prototype) of the product under development, whereas agile development iterates on the

actual product. This observation leads to the discussion of the first point of difference as listed in table 2.1, and the point debated by Beck and Cooper [97].

Interaction designers understand that much of the work driving the development of the product takes place before the product is implemented. Therefore, interaction designers assume that all the requirements are known up front and consequently, their design incorporates all the tasks that users will require when using the product. This can be referred to as a holistic approach [37]. Agile development, on the other hand, encourages developers to begin implementation at the start of the development project, acknowledging that every requirement will not be known up front, and that new requirements may emerge during the software development effort — hence the need for an iterative approach, to allow new requirements to be built into the product. This is known as *incremental development* [65].

The final difference listed in table 2.1, relates to the composition of software development teams. Interaction design skill is understood to reside with interaction designers, who are trained in the techniques of interaction design discussed in section 2.4.2. Developers who are skilled at coding software are not necessarily also trained in interaction design techniques. While possessing both skills is not an impossibility, anecdotal evidence suggests that it is rare to find people who do both well at the same time. Cooper compares developers, unskilled in interaction design, yet doing interaction design, to carpenters designing houses:

If carpenters designed houses, they would certainly be easier or more interesting to build, but not necessarily better to live in [35, p23].

Agile development prefers teams composed of generalists [70, 93]. Although the agile team members have roles, these are not seen as rigid and team members contribute in the best way they can.

Interaction design and agile development have much in common — an appreciation for the importance of evaluation of customer satisfaction and how an iterative approach is the best way to accomplish this — however

even the superficial similarities revealed some differences when probed further.

2.5.3 Related studies

Despite the differences, there are notable studies of teams successfully combining interaction design and agile development:

Jeff Patton describes how interaction design and agile development can work together. The agile development activities were directed by the artifacts created by the interaction design process (Constantine's Agile Usage-Centered Design approach [33]), which was seen to benefit the overall development effort [109, 110].

Beyer, Holtzblatt and Baker [13] show how the Contextual Design approach to interaction design naturally fits with agile development. This approach advocates understanding end users and the scope of the product under development before development begins, with an iterative interaction design process taking place before implementation begins.

Lynn Miller describes experiences in managing projects where the interaction design was of critical importance to the software [96]. The approach sees the interaction design and the programming done in parallel, but one iteration out of phase. In this way, the interaction designers were doing detailed design for the iteration that the developers would do next, and doing evaluation of the iteration that the programmers did last.

Desirée Sy describes a similar approach [137]. During the team's adoption of an agile development process, the user-centered design activities were adjusted and improved. In this report, the interaction designers designed the UI, such that they were designing features that developers would be implementing in the next iteration, while ensuring that the implemented interaction design did not 'drift' away from the specification in the current iteration

Chamberlain, Sharp and Maiden [23] use a field study to ground their

introduction of a broad framework for how interaction design and agile development can work together. In particular, their study shows, and their framework explains, how the general values and practices typical in interaction design and in agile development are quite similar and can assist teams in working together, but that efforts must be made to ensure balance, appropriate resource management, participation and a coherence of purpose.

McInerney and Maurer [93] conducted a mini study of user-centered design specialists on agile development teams. This study identified possible implications that agile development could have on UI design, namely, that UI design may have to focus on a small piece of the whole application at each iteration, that UI design may be more of a team effort and that the UI designer is continually involved in discussions during development. The user-centered design practitioners in this study were very positive about working alongside agile teams.

2.6 Summary

This chapter highlighted the characteristics of the software development environment, which agile development attempts to address. eXtreme Programming, Scrum and interaction design were described, along with the consideration of the place of interaction design within software development. Interaction design and agile development were compared to highlight the similarities and differences that may affect the way in which they are combined in practice. Finally, studies in the literature of agile teams combining agile development and interaction design were described.

Chapter 3

Research Method

This chapter explains the research method that was used for this study. We begin with discussing the appropriateness of using the grounded theory approach and why we chose it. We then give a brief introduction to grounded theory, along with details about the researcher and research perspective. The participants are introduced in section 3.4. We also illustrate the data gathering and analysis methods and discuss aspects of the study that affect the validity and reliability of the results.

3.1 Why is grounded theory appropriate?

There has been little investigation or discussion on how interaction design and agile development work together, and therefore little guidance for our research question: “How do real-world agile teams combine interaction design with their agile development activities?” Creswell states that

[..] the qualitative study approach is considered an appropriate method when little is known about the phenomenon under investigation and the concepts are immature due to lack of theory and previous research and a need exists to explore and describe the phenomena [39, p145].

With no established theories or practices in this area, we decided a qualitative approach would be most suited to this research. Further, qualitative research methods are “designed to help researchers understand people and the social and cultural contexts within which they live” [94], whereas quantification of textual data loses sight of these contexts and phenomena [74].

Our research being in the context of software development, where agile methods and interaction design are seen to be strongly people-oriented [28, 54], we followed the suggestions of Duvall [44] and Lings and Lundell [89], who identified the grounded theory approach as appropriate for studying the socio-technical aspects of software development.

Due to limited time for this study, a full grounded theory study where theoretical saturation is reached in every core category, was not attempted. Similar to Charmaz’s observations of many grounded theory researchers, we concentrated instead on building a rich description of the phenomena under study [24] — in our case, interaction design and agile development in practice. Instead of testing or confirming existing theories, we concentrated on defining relationships in the data as they emerged from the data.

There is a growing body of literature where grounded theory is being employed in studies about the people-aspect of software:

1. **Grounded theory and agile development research**

Recent studies by Whitworth and Biddle [145, 146] used the grounded theory approach to investigate social aspects of agile teams. The XP Customer Role has been investigated involving a grounded theory analysis by Martin, Biddle and Noble [91].

2. **Grounded theory and methodology research**

Hansen and Kautz concluded that the use of grounded theory can provide valuable results in research regarding information systems development methodology [68]. Cockburn employed grounded theory techniques in his PhD research addressing the relationship be-

tween people and methodology [30].

3. Grounded theory and human-computer interaction research

Singh, Bartolo and Satchell applied grounded theory to discovering a software system's user requirements from a social perspective [129], while Swallow, Blythe and Wright used the grounded theory approach for analysing user experience evaluations of smart phones [136].

3.2 Grounded theory

We adopted *grounded theory* — a qualitative research method that promotes the generation of theory from data [64, 113, 25]. *Theory* here “constitutes an integrated framework that can be used to explain or predict phenomena” [132]. The theory is said to ‘emerge’ from the analysis of data obtained from interviews, observation sessions, etc., and is, therefore, ‘grounded’ in reality. Glaser further explains that grounded theory is “the generation of emergent conceptualizations into integrated patterns, which are denoted by categories and their properties” [63]. These patterns are achieved by rigorously applying the analysis techniques which involve the analyst continuously asking questions such as Who? When? Where? etc. and making comparisons of categories.

Once the research questions for the study have been established, the *coding* of the data begins. Coding allows the tabulation, interpretation and analysis of data [40] and Strauss and Corbin distinguish between three types: *open*, *axial* and *selective coding*. Open coding allows the researcher to break the data up into discrete concepts that are named based on the phenomenon identified. These are referred to as ‘categories’. Axial coding is a process of connecting these categories and establishing cause-and-effect relations between them. During selective coding, the relations are refined and a theory evolves [132].

Another essential aspect of grounded theory analysis is *memoing* — recording ideas about relations and other coding decisions throughout the data collection and coding process. Memos play a vital role in structuring the categories and, eventually, the theory itself. Glaser [62] and Strauss and Corbin [132] stress the importance of performing memoing throughout the entire study.

It is worth noting that the steps described here are not required to be performed sequentially. At times during the analysis, open and axial coding may be taking place at the same time, while memoing is ongoing from the start of the study to its conclusion.

3.2.1 Theoretical sampling

As the study unfolds, based on the categories that emerge from the data, the researcher determines what data to collect next. This is known as *theoretical sampling*. Glaser explains:

Theoretical sampling is the process of data collection for generating theory whereby the analyst jointly collects, codes, and analyzes ... data and decides what data to collect next and where to find them, in order to develop ... theory as it emerges. This process of data collection is controlled by the emerging theory, whether substantive or formal [61, p45]

Generally, the categories with the greatest explanatory power are pursued in subsequent data collection, so that they may be further developed and refined [108].

3.2.2 Theoretical saturation

Theoretical saturation is a point in data analysis where

[...] no additional data are being found whereby the (researcher) can develop properties of the category. As he sees similar instances over

and over again, the researcher becomes empirically confident that a category is saturated ... when one category is saturated, nothing remains but to go on to new groups for data on other categories, and attempt to saturate these categories also [64, p65].

At the point of theoretical saturation in the core categories, theoretical sampling stops.

3.2.3 Role of the literature

In grounded theory, the role of existing literature is two-fold:

1. To focus the research questions; and
2. Provide a source of data to compare the emerging theory to.

The relevant literature helps to formulate the research questions and frame them in a way that they give the researcher the flexibility and freedom to explore the phenomenon in depth [61, 132].

Comparing concepts from the data with the literature for conflicts and agreements, further enhances the findings. When the emerging data disagrees with existing literature, researchers have the opportunity to establish unique features of already existing phenomena in their field of research. When the emerging data is similar to the discussions in the existing literature, this may tie together underlying similarities in phenomena not usually associated with each other [46].

3.3 The researcher

Of utmost importance has been to ensure that it is the

[..] participant's perspective on the social phenomenon of interest that should unfold as the participant views it and not as the researcher views it [90, p101].

The life experience of the researcher will, inevitably, influence the researcher's interpretation of the data [39], therefore my professional and educational background is explained in this section.

I hold a BSc(Hons) in Computer Science and a BCA in Economics and Econometrics, both from Victoria University of Wellington, New Zealand. Courses taken for the computer science degree included basic and advanced programming techniques, software engineering and user interface design. My honours research focused on user interface design and its relation to the theory of Peircean semiotics. I have experience developing a database application for the editor of the Foreign Policy Journal, to manage submissions to the journal and other editorial responsibilities. During the development of this application, I was the sole developer. I had complete freedom in terms of the interaction design and development methodology. My most recent employment was as an assistant developer on the Research and Development team of a large international market research company. In this position I developed XML-based templates for generating and charting survey results, which required a good knowledge of XML and statistical analysis techniques. I did extensive programming in Visual Basic and .NET developing customised reporting solutions for statistics researchers and developing automated solutions for manual reporting processes. For two years I have been an active member of the agile software development community.

3.4 The participants

This study involved thirteen participants working on eleven projects in different countries around the world — Canada, USA, Finland, New Zealand and Ireland. Participants were members of co-located agile software development teams ranging in size from five to twelve, including interaction designers. Teams in this size range would be considered small to medium. On average, participants had two years experience with combining agile

development and interaction design and were between three months and three years into their projects at the time of the interviews. We interviewed participants on XP and Scrum teams and therefore, when we refer to agile development in this thesis, this is based on the notions of agile development in terms of XP and Scrum. Table 3.1 summarises the information about the teams and their projects, described below:

P1

The first two participants, working on project P1, develop and market web-based software to support IT managers and development professionals. Based in the United States, P1 is an XP project, where the team consists of ten engineers and one product manager/user interface designer. We interviewed the Engineering Manager and Product Manager/User Interface Designer from P1. At the time of the interviews, P1 involved redesigning and enhancing an existing product. Their Product Manager/User Interface Designer described the project as follows:

“There are several features we added and several things that we wanted to do with the product. And one thing we noticed was that performance was really bad, it was built upon a really terrible code base. It was just all hacked together. User interface was terrible and, you know, the user interactions were very cumbersome. So we decided from that generation of product that we were gonna rewrite and start from scratch. We took everything that we wanted out of the old product and built it the way we really wanted the product to end up ... During this process we adopted the XP methodology.” — *Product Manager/UI Designer, P1*

P2

P2 is based in Ireland. Participants on P2 develop and sell stand-alone software to support wealth management. P2 involves an XP team that

includes four engineers, one domain expert/on-site customer and two interaction designers. Their Project Manager and one of the Interaction Designers provided interviews and described their project and its status at the time of the interviews:

“We’ve been focusing on sort of single-user client systems.” — *Project Manager, P2*

“There are smaller projects as well and we have our bigger, our overriding kind of application building, which is our wealth planner application, which we’ve been working on for two years and we have our first customer for that now and we’re releasing that in a few weeks. And we’re a certain of a way through developing. It’s quite big, so it’s an ongoing project.” — *Interaction Designer, P2*

P3

P3 is based in New Zealand and its XP team develop software that controls fruit sorting machines. Their team consists of five developers, one of whose main interest is interaction design. We interviewed two developers on this team, including the developer with an interest in interaction design. Their main project was described as follows:

“Our central control program controls fruit sorting machines and does all the controlling of the machine and talking to all the sensors that are gathering information about the fruit’s colour, and things like that, and gathering this all together and presenting that information to our customer, so our customer can make choices about how they’re going to sort their fruit into different packages and what not.” — *Developer, P3*

P4

At the time of the interview, the participant (an interaction designer) on P4 was employed by a software consulting company based in Finland, employing Scrum to develop

“A [web-based] system to manage teaching in the ... just a bit lower than university level school, meaning that with this system they decide what courses are given at what point of time during the semester and who’s going to teach them. Kind of try to optimize this so that the workload for students would be manageable and the workload for the teachers would be manageable. And they also keep their curriculums in the same system, so they have all this information available ... so they can kind of instantiate for the next school year what courses they’re going to give ... I don’t think that it’s going to end very soon because it’s really huge and a very important piece of software and it has been going on now for one and a half years.” — *Interaction Designer, P4*

P5

P5’s team consisted of five developers and was part of the same company as P4 in Finland. We interviewed the team lead/developer on the team, who worked across several projects:

“I’m currently working on one of our bigger clients where we have a team of about five developers working across different projects. We do maintain a constant presence with our customer doing smaller day to day things.” — *Developer, P5*

P5 worked on developing a new web-based application, using Scrum.

P6

The developer we interviewed on P6 is part of an XP team in a small company in Canada that develops phone communication systems which integrate with other messaging software. Within the team there were no definitive roles, other than programmer, however, the participant we interviewed assumed most of the responsibility for interaction design. The participant described the customer requirements and the team’s adoption of XP for the project as follows:

“We had a large customer that deployed their system in two physical locations providing several hundred phones ... So they would come and say, ‘We need our software to do this, we need the menu to look like that, we need to have these features’ ... and then it is up to systems engineering to figure out how to give them that. It is kind of a very, very rudimentary provision of services ... we weren’t doing anything remotely approaching agile or extreme, and we introduced a lot of the programmer-centric principles of extreme programming – test-driven design, some pair-programming ... We’ve introduced user stories, we’re going to be doing iterations soon. We are tracking estimates and we are tracking velocity and getting an understanding of what that means, and understanding what the team can complete.” — *Developer, P6*

P7

We interviewed two developers (Developer1 and Developer2) on P7, another project based in Canada. The team on this project employed the Scrum method and their web-based project was described as follows:

“We’re looking at a business tool that’s used by business users who are building marketing campaigns for running on their websites. It’s not a new product, it’s a new aspect to our product that we work on. It’s not like replacing an old tool. It’s a new functionality that wasn’t there before.” — *Developer1, P7*

When questioned about the nature of the roles in the team, Developer2 responded:

“They’re grey. We do have them but the team is dynamic enough that day in day out they kind of know what their responsibilities are ... depending on the exact set of requirements we’re trying to deal with ... So we have roles ... but in reality it’s mostly about what needs to get done and who is the best person to do that.” — *Developer2, P7*

P8

The participant from P8 was the project manager for three XP projects in Canada and discussed aspects of all three during the interview. For convenience these projects will be collectively referred to as P8, however, to aid the discussion here, we will distinguish between them as P8.1, P8.2, P8.3. A brief description of the three follow:

“[P8.1] has to do with companies or individuals that want to create a business ... There’s a current system that exists ... and it’s been out there for about seven years. It’s got somewhere about 900,000 transactions per year. The other project that I’m dealing with is [P8.2]. So, people who are lobbying for certain causes have to register themselves and have to follow certain regulations whenever they talk to people that are part of the government ... So the number of transactions ... from about 6000 transactions per year it’ll go to about 60,000 to 100,000 transactions per year ... The third project that I’m working on is [P8.3], they provide insurance on small loans for small businesses ... And they have something like about 15,000 registrations, or loans, that they actually cover per year, and they expect that to increase because of new products that they’re taking out.” — *Project Manager, P8*

For all three projects, the ratio of interaction designers to developers on the XP team

“[Depends] on the size of the development team, how fast the client wants to move. We figure it takes about one interaction designer for two developers. So, we kind of have a 50% ratio there ... for example, the [P8.1] project we have six developers, one architect, two interaction designers and one dedicated QA person and plus a dedicated project manager ... the [P8.3] have a small team of one database specialist, ... one interaction designer and two to three developers ... [P8.2], we have two interaction designers, four developers.” — *Project Manager, P8*

P9

The final participant was a developer on P9, a project based in the US. They developed web-based software for conducting on-line surveys, as well as reporting the results. The participant explained:

“Web data collection framework... [which] allows for a group of employees within the company that have minimal programming knowledge to create and deploy questionnaires to our clients... It provides our clients with the ability to track survey participation when a survey is live, conduct e-learning, or take additional questionnaires, download reports and or create reports based on collected data.” — *Developer, P9*

The roles in the team were well-defined and were divided into a project manager, a group responsible for requirements gathering, developers and a data visualisation specialist. The interaction design was the responsibility of the data visualisation specialist, whose role was described as:

“Someone not dedicated to the [development] team, but part of a general pool of data visualisation experts. The lead developer collaborates closely with them up front to explain the deliverable, and provide some fundamental UI constraints that need to be adhered to. The outcome from this person is the UI — typically as a set of images, or storyboard.” — *Developer, P9*

3.5 Conducting the study

According to Fontana and Frey, interviewing is “one of the most common and powerful ways in which we try to understand our fellow human beings” [51]. We chose to use semi-structured interviews with questions that were open-ended, could be fully expanded at the discretion of the interviewer and interviewee and could be enhanced by further probing [121, p149].

Our interviews were conducted with volunteers from real-world software development teams. These interviews were voice-recorded and then transcribed line-by-line. The analysis of the transcripts followed the grounded theory approach, which included open coding, axial coding, selective coding and memoing. This process is explained in section 3.5.3. During the analysis process, the concepts related to our initial research questions emerged from the data. Through constantly comparing between concepts as they emerged, we attempted to discover the relations between them and in so doing, build a description of interaction design and agile development in practice.

3.5.1 Participant recruitment

The participants were recruited through networking with members of the agile software development community, as well as approaching software development companies who practiced agile development methods with significant interaction design. Participation was voluntary and secured on an individual basis. Participants signed confidentiality agreements to protect their identities and all identifying information has been removed from the illustrating quotes. The information provided to participants is included in appendix C and includes the ethics approval document, which was required by the university's Human Ethics Committee (HEC).

3.5.2 Interviews

All interviews were conducted at a location which was most convenient both for the researcher and the participant, and was agreed upon on a case by case basis. The interviews were face-to-face and one-on-one, except for the interviews with P2, which was conducted over the telephone with both participants present, and P9, where the participant responded in writing. The questionnaire was intended to probe the participant about their process of combining interaction design with agile development, and

the challenges they faced. The aim was to inquire into how and when certain activities, relating to interaction design, were performed with respect to the agile development activities. To address our research question, our interview questions centered around

- How the interaction design and agile development activities were organised at different stages of the development effort,
- How the teams obtained feedback about their interaction design within the structure of the agile development iterations and incorporated that feedback into subsequent development and
- How the interaction designer role related to the agile development team.

The questions were open-ended. The questions included in the questionnaire were obtained from analysing existing literature addressing interaction design and agile development, and formulated with the research questions in mind. The first interview questionnaire is attached in appendix B.1. According to the principle of theoretical sampling (section 3.2.1), it is considered good practice to revise and adapt the interview questions for subsequent interviews, based on the results of the analysis of the first interviews. The revised interview questionnaire is presented in appendix B.2 and pursues the core categories relating to interaction design activities that take place before development begins, how they fit into the structure of the agile development iterations and the place of the interaction designer on the development team.

The interviews were voice-recorded with the participant's permission and then transcribed line-by-line. All interview transcriptions were returned to the respective participant for their feedback regarding the accuracy of the transcription, then corrected if necessary. The interviews lasted approximately 44 minutes on average.

3.5.3 Data analysis

Coding the data The first step once the interviews had been transcribed and verified to be correct by the participants, was to code the data. It was found that during open coding, the context in which the sentences were spoken was preserved when coding the data sentence by sentence — even a collection of sentences. This turned out to be helpful in the axial coding stage. A list of codes was maintained in order to lighten the memory load of the coder. During open coding codes were added to this list as they emerged. Once all the interview data had been assigned codes, duplicate codes or codes with similar meanings could easily be identified by referring to the list of codes. Duplicate codes were eliminated and similar codes were merged into a single code. This list contained 402 codes and a random extract appears in figure 3.1. Extracts from the interviews were entered into a database application using the form depicted in figure 3.2. Discrete fragments of the interview transcriptions were entered into the ‘Excerpt’ field and the codes associated with that fragment were entered into the ‘Category’ field. In this way more than one code or category could be associated with one fragment, while information about the participant and the location of the fragment in the interview transcription document could be preserved. There were definite advantages in storing the interview and coding information in the form of a database, namely that the database application could assign unique identifiers automatically to interview fragments and could filter entries according to criteria and then export those entries in a table form, which was found to be useful for the axial coding process.

Axial coding organised the related codes into a number of higher level conceptual categories and established relationships between them. The context provided by the sentences were especially helpful for considering categories in relation to other categories and discovering the nature of those relationships.

Memoing In figure 3.2 the field called ‘Comment’ was used to note any

ideas about possible relationships, reflections and tentative hypotheses that occurred during analysis. Another example is shown in figure 3.3.


3.5.4 Theory development

As a means of capturing the development of theory, a separate document was created where the categories of each interview were incrementally added and the memos incorporated — starting with the structure provided by the categories of the first interview and adapting the existing structure to take the categories emerging from subsequent interviews into account. The participant quotes were selected at this time as illustration of the main categories that emerged.

3.6 Reliability and validity of the study

There are many factors that can threaten the validity and reliability of qualitative data [95]. Validity in grounded theory studies are achieved by adhering to the prescribed grounded theory techniques as outlined in section 3.2, particularly through constant comparison. According to Glaser, findings that are grounded will also necessarily be valid [63]. Further, Glaser proposes two criteria against which the quality of an emergent theory should be evaluated [61]. The first is that the theory should fit the data, which means that the categories should emerge from the data — without the researcher forcing data to fit conceptual categories that can not be found in the data. The second criteria is that the results explain the core problems and processes in the relevant area, that may be useful for practitioners in that area.

The following points outline the risks we identified and the measures to address each risk to increase the accuracy of the data and reduce researcher bias:



	A
1	Codes
76	Developers try to understand customer's domain
77	Developers write user stories from ideas
78	Development process * *
79	Development tools *
80	Development tools built up over time
81	Development tools must be cheap
82	Development tools must be easy to use
83	Different teams working on the same system
84	Disadvantages of ignoring interaction design
85	Disadvantage of not having holistic view
86	Discussion
87	Discussion and debate is good
88	Domain expert
89	Domain expert tests every few days
90	Domain expert develops acceptance tests
91	Domain expert is also the customer
92	End user * *
93	Estimates *
94	Evaluating interactions
95	Evaluating usability
96	Everything in the system is linked to user tasks
97	Existing system
98	Experience of developers will improve usability
99	Expert review
100	Feedback gathering
101	Feedback not direct to developers
102	First release
103	Front end difficult to apply XP principles - simple design & refactor
104	Front end difficult to test
105	Functionality over usability

Figure 3.1: Code list

The screenshot shows a window titled 'frm_InterviewAnalysis'. It contains a form with the following fields:

- ID:** 17
- Company:** P2
- Excerpt:** They come from our workshop. We gather the user stories via that.
- Category:** Workshops, understanding users, user stories, ui requirements
- Comment:** (empty text area)
- Speaker:** Interaction designer
- Lines:** 45

At the bottom, there is a record navigation bar showing 'Record: 14 of 463' with navigation buttons.

Figure 3.2: Coding

...ing - ... implementation ...

- [ID=295][P3] Even if you can't evaluate with a user, I think it still gives you an idea about whether it will work or not. At least you can see potential problems based on experience, standards, and conventions.

[up front ui design -> gives an idea of whether it will work or not, see potential problems -> even if testing with users not possible -> based on -> experience, standards, and conventions]

- [ID=127][P1] I've talked to other people - they told me the same thing - that UI designers hate this thought. UI designers like to think, like, this is the overall thing, this is the whole architecture, this is everything. And, just, like, doing a little bit at a time, and not thinking about the rest it's something. I mean, [interaction designer] had a lot of problems initially but we really had to tell him a couple of times, "This screen that you did for us, this is for ten user stories, all we implement is two." You can't just leave these elements in there and just not make them work and he hated it at the beginning.

[UI designer hates -> not having overall view, working iteratively, concentrating only on a small set of features at a time -> problems in the beginning][development team wanted a UI with all implemented features (working at all times)]

- [ID=129][P1] Initially what you get is, pretty much for the whole release that is, oh, ok, these are all the user stories that we need or that we are planning to implement, so I make a UI for all these fifteen user stories. And then we started cutting out features, changing features along the line, so I think that's something that we have finally convinced him to do, so don't start with this holistic view when we know that we will have to change it anyway. To make it in small steps it actually saves you time. As long as you have your overall, kind of, goal in mind, like this was the design that we thought for this whole feature, so kind of the UI metaphor in this case. It's something that you don't write down

Comment [29]: Experience, standards etc. that other teams talk about too.

Comment [26]: XP design strategy - all tests running. So having unimplemented features sitting in the UI meant extraneous elements. XP principle: simple design. See book extreme Programming explained (ch 17).

Figure 3.3: Memoing

- **Risk:** Small, unrepresentative sample

Measure: Although our sample was not large, we attempted to include a variety of participants. Our participants were based in different countries around the world, developing software for a range of different domains on a variety of project sizes. We interviewed professional software developers instead of university students. Compared to other grounded theory researchers who have used six participants in their studies [5, 84, 95], this study interviewed thirteen participants. The experience of developing grounded theory for this thesis has been akin to that of Martin and Turner who state that “By the time three or four sets of data have been analysed, the majority of useful concepts will have been discovered” [92]. Participants interviewed were either on XP or Scrum teams. The agile development methods known as XP and Scrum adhere to the values of the agile manifesto¹ and, therefore, the findings of this research should be transferable to similar development methods that adhere to these values.

- **Risk:** Inaccurate recording of data

Measure: The interviews were voice recorded to ensure accuracy and completeness of the data. The interview transcriptions and findings were then verified with participants.

- **Risk:** Forcing the data

Measure: We took care for the theory to emerge from the data, so that it was *grounded*, rather than forced to fit preconceived frameworks. This was achieved by rigorously adhering to the grounded theory techniques.

- **Risk:** Interviews only

Measure: Not only the interview transcriptions, but also the results

¹<http://www.agilemanifesto.org>

of the analysis were returned to the participants for their review. The results of this study were presented at the major international agile development conferences where the results were favourably accepted by practitioners and academics.

Table 3.1: The participants

Team	Agile	Country	# Participants	I
P1	XP	USA	2	P
P2	XP	Ireland	2	D
P3	XP	New Zealand	2	D
P4	Scrum	Finland	1	D
P5	Scrum	Finland	1	D
P6	XP	Canada	1	D
P7	Scrum	Canada	2	D
P8.1	XP	Canada	1	P
P8.2	XP	Canada	1	P
P8.3	XP	Canada	1	P
P9	XP	USA	1	D

* The project manager was the

Chapter 4

Introduction to results

In this chapter we describe the large-scale features that characterised our participants' approaches to combining interaction design activities with agile development activities. The introduction given here provides the context for the discussion throughout the rest of the thesis, with the substantiating quotes from the participants presented in the following chapters.

The motivation for this research was to aim for a better understanding of how interaction design and agile development were being combined by agile teams in practice, including how practitioners were performing the interaction design activities, relative to the agile development activities.

Through the participant interviews, we discovered that agile teams were broadly following similar patterns of combining interaction design and agile development, irrespective of their agile development methodology. From the data it became evident that the teams organised their activities according to two stages of the development effort:

- Before implementation began and once the development project was considered started, or up-front, and
- Once implementation had begun.

Within these stages, approaches to the design and implementation of the

interaction design emerged as the combination of a *design strategy*, in which the interaction was designed, and an *implementation strategy*, in which the interaction design was implemented as functional software.

There were also insights into the value of doing some interaction design up front (before implementation begins), despite popular opinion that doing this contradicts agile values. We investigated the effects that the iterations of the agile development process and the interaction design process had on each other, as well as the role of the interaction designer on agile teams.

4.1 Stages of development

From the data we observed two main stages into which the software development activities of our participants could be divided. The first was the stage before implementation began. At this time, the development project was considered started, however, no coding activities had begun. We refer to this stage as ‘Before Development Begins’. The activities taking place at this stage were aimed at researching the scope of the product under development and researching the potential users. The second stage of software development was the implementation stage. At this time the implementation activities, activities related to coding the software, dominated. These stages were observed in the case of all teams.

4.2 Interaction Design Approaches

Within the stages of development distinct strategies for designing and implementing the interaction design emerged. We refer to these strategies as *Interaction Design Approaches*. The interaction design and agile development activities of the teams grouped into four different approaches, which were combinations of strategies for dealing with the design of the interaction design, and strategies for its implementation as functional software.

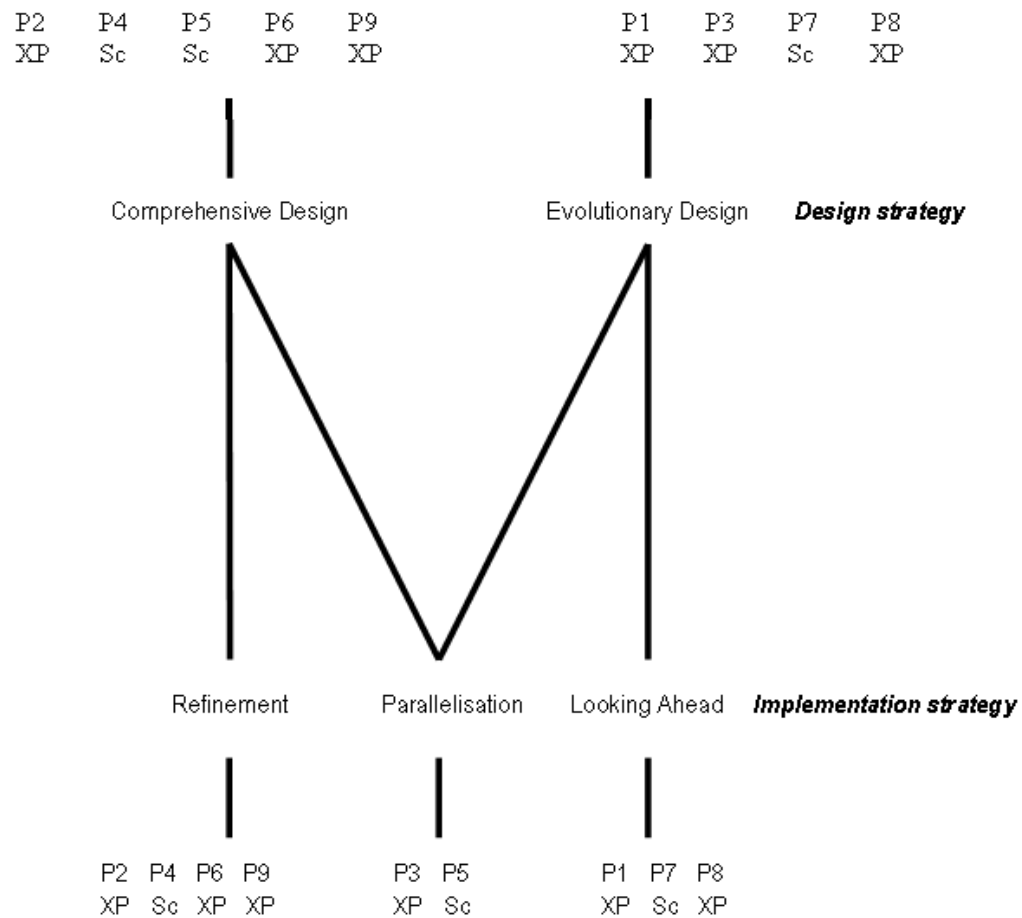


Figure 4.1: Teams and their interaction design approaches

We represent these approaches in figure 4.1. There emerged two strategies for dealing with the design aspect of the interaction design: *Comprehensive Design* and *Evolutionary Design*. We refer to these strategies as design strategies. There emerged three strategies for dealing with the interaction design once implementation had begun: *Refinement*, *Parallelisation* and *Looking Ahead*. We refer to these strategies as implementation strategies. The four combinations of design and implementation strategies are represented in figure 4.1 as:

1. Comprehensive Design with Refinement,
2. Comprehensive Design with Parallelisation,
3. Evolutionary Design with Parallelisation, and
4. Evolutionary Design with Looking Ahead.

4.2.1 Design strategies

As explained in section 2.4, user research is traditionally an interaction design activity and all participants performed this interaction design activity up front. Where the teams diverged in their design strategy was in the amount of interaction design completed before implementation began and the way design and implementation of the interaction design progressed once implementation had begun.

One group of participants completed a high percentage of the interaction design up front, which was implemented during the agile development iterations. These teams understood interaction design to be a separate process to agile development, and that the two should occur in sequence. Teams who subscribed to this view completed their UI design such that it was a representation of the entire system under development. When a complete design is created before implementation begins and that design is used to guide the rest of the development activities, practitioners refer to this as doing Big Design Up Front (BDUF). Practitioners intend BDUF to mean any comprehensive design done up front — whether for coding or interaction design. In this thesis, we will refer to the BDUF strategy of interaction design as *Comprehensive Design* to refer to the strategy where a comprehensive design is completed for the interaction design, but not for the coding aspect of the software. Figure 4.1 shows that teams P2, P4, P5, P6 and P9 followed the Comprehensive Design strategy.

The other view held by participants was that the interaction design and agile development processes should be merged into one process. This

view considers the ideal combination of the processes as one where interaction design blends into agile development and the interaction design activities are interleaved with the agile development activities throughout development. Participants who subscribed to this view produced a UI design that only implemented the features from previous iterations and the features that had been selected for a set number of iterations ahead. These teams performed initial design activities in order to get started, but did not complete a comprehensive design before implementation began. Instead, the design was allowed to evolve during the implementation stage. When a system is designed in this way, Fowler refers to this as Evolutionary Design [52]: “Essentially evolutionary design means that the design of the system grows as the system is implemented,” and in a conversation with Bill Venners, “You begin by coding a small amount of functionality, adding more functionality, and letting the design shift and shape” [142]. In this thesis, we will adhere to Fowler’s term and refer to the strategy where an interaction design grows as the system is implemented, as *Evolutionary Design*. Figure 4.1 shows that teams P1, P3, P7, and P8 followed the Evolutionary Design strategy.

4.2.2 Implementation strategies

As explained in chapter 2, agile development iterations consist of activities that can be grouped into planning, implementation and evaluation activities. Although each team carried out planning, implementation and evaluation activities, the way the team organised the sequence of these activities with interaction design activities, depended on factors such as the amount of interaction design completed up front and the degree of separation that could be attained between development of the front end and the back end.

Based on the participants’ responses we identified the implementation strategies by asking the following questions:

- What work occurs before implementation begins?
- What drives development?
 - What are the plans for the iteration based on?
 - What do developers refer to when implementing the UI?
- What constitutes the work of the interaction designers when the developers are doing implementation work?

Based on the answers to the above questions we could distinguish between three implementation strategies — *Refinement*, *Parallelisation* and *Looking Ahead*.

Refinement

The most popular implementation strategy among our participants who followed the Comprehensive Design strategy, was what we refer to in this thesis as *Refinement*. This was followed by teams P2, P4, P6 and P9. The term for this implementation strategy is taken directly from the participants' own words for their descriptions of their activities. For example, the developer from P6 talks of the agile development iterations "as a refinement process."

For teams who followed the Refinement strategy, interaction design was seen to occur before implementation began, followed by the agile development process. Developers matched their implementation to the UI design specification during the agile development iterations and changes to the UI design were expected to be mainly due to implementation issues encountered by the developers. These were usually minor design changes required to implement the design with the technology being used to develop the product. Thus, the comprehensive UI design created up front was refined during the iterations, successively transforming it into an 'implementable' interaction design.

Parallelisation

Only two teams in our study followed the strategy we will refer to as *Parallelisation*. Both teams had a different design strategy. P5 followed Comprehensive Design and P3 followed Evolutionary Design. Again the term for this strategy is taken from the participants' responses. For example, the developer from P5 explaining that interaction design "is done in parallel to the development."

Applying the Parallelisation strategy appeared to be independent of whether or not a comprehensive interaction design had been created up front, and more related to whether the product and the implementation technology allowed a separation between the front end (UI) and the back end (system code). Parallelisation of the interaction design and agile development relied on the underlying system code remaining unaffected by interaction design changes and vice versa. As the developers iteratively implemented the system, the UI was iteratively designed and evaluated separately. The interaction design was not seen to drive development. Instead, user stories or small chunks of formulated tasks taken out of a requirements specifications document determined the work to be done in the development iteration. Due to the limited reliance of the two processes on each other to progress, the number and timing of their iterations were also independent of each other. As a result, the interaction design and implementation required synchronisation at some point during development for evaluation of the whole system to take place.

Looking Ahead

Three out of the four teams in our study who followed the Evolutionary Design strategy, followed what we refer to in this thesis as the *Looking Ahead* strategy. These were teams P1, P7 and P8. The term for this implementation strategy is also taken from the participants' own words for their descriptions of their activities. For example, the Project Manager from P8

explains that the interaction designer de-risks the implementation process “because they’re looking at it [the product] from a little further away.”

The Looking Ahead strategy was characterised by the interaction designers creating a design for a fixed number of iterations ahead (usually one to two iterations ahead) of what the developers were implementing in the current iteration. So, interaction design and agile development were occurring simultaneously, however, as the agile development iterations progressed, so the UI was iteratively designed to stay ahead of the implementation. For these teams, after any number of agile development iterations, the UI design covered only the features that had been selected to be implemented up until the iteration they were designing for. Unlike the Refinement strategy, where all the interaction design activities took place before implementation began, the Looking Ahead strategy allowed more interaction design activities to be interleaved with the development activities. For example, the interaction designers could iteratively evaluate UI designs with users before they were implemented in subsequent iterations. In this way, interaction designers were looking ahead of the developers at what issues user evaluations were elucidating about the system under development, before those issues became part of the implemented system.

Combinations with design strategies

Apart from the teams who followed the Parallelisation implementation strategy, the rest of the teams’ implementation strategies, were dependent on their design strategies. Figure 4.1 shows that teams who followed the Comprehensive Design strategy, only followed Refinement or Parallelisation as implementation strategies, whereas teams who followed the Evolutionary Design strategy, followed Looking Ahead or Parallelisation. None of the teams combined Comprehensive Design and Looking Ahead nor did they combine Evolutionary Design with Refinement. While the combination of Evolutionary Design with Refinement suggests that it would be

difficult to carry out in practice, combining Comprehensive Design with the Looking Ahead strategy would certainly be possible, albeit a great waste of resources spent on completing a Comprehensive Design in the first place.

4.3 Feedback and change

Regardless of whether the teams were XP or Scrum teams, the agile development iterations were found to be opportunities for obtaining feedback about the interaction design and incorporating changes back into the interaction design. We examined how teams obtained feedback about their interaction design within the structure of their agile development iterations and how that feedback was incorporated into subsequent development. Four strong themes emerged from our data (chapter 7 details these points):

1. Development iterations drive usability testing
2. Usability testing results in changes in development
3. Iterating with working software brings insights
4. Iteration planning affects interaction design

These points show a coherent picture of how interaction design and agile development can work together for considerable advantage. Iterations can be seen as driving usability testing, which in turn affects subsequent development. Teams discovered insights into design and technology limitations as they implemented the system and iteration planning affected interaction design in terms of what UI features were implemented in an iteration.

Feedback and change was not only encountered in the product under development, a re-occurring theme in the interview data was the experimentation and adjustment of the development process itself. All teams

were constantly calibrating their development process to better meet the end user's goals and increase the overall productivity of the interaction designers and developers on the project. Combining interaction design and agile development activities during the development effort was seen as a learning experience that teams were becoming better at over time.

4.4 Interaction designers on agile teams

The final research question for this study concerned the interaction designer role. The participants' emphasis on the importance of creating usable software and their acknowledgment that interaction design requires skills that are different from programming skills, emerged from the interview data. The interaction designer role contributed these interaction design skills to the agile team, which was understood to add great value to the overall development effort. A major insight obtained from the participants was that interaction design in an agile context is considered a collaborative effort between interaction designers and developers. Whether interaction designers created a comprehensive UI design before developers began implementation, or only a subset of the final UI design, implementation activities provided insights into the interaction design that required agreement between the developers and interaction designers before the implementation or the design could be changed. Another major insight was that the amount of collaboration between interaction designers and developers depended on whether the teams considered themselves composed of generalists or specialists. Teams of generalists encouraged feedback from all team members, whereas teams composed of specialists required each role to limit feedback on issues outside their area of expertise.

4.5 Publication of results

The results discussed in chapters 5 to 8, have been published and presented at various conferences. A combination of the results in chapters 5 and 6 were presented at the 8th International Conference on Agile Processes in Software Engineering and eXtreme Programming, held in Como, Italy, on June 18–22, 2007 [50]. The results from chapter 7 were presented at the Agile 2007 Conference held in Washington, DC, USA, on August 13–17, 2007 [48]. The results from chapter 8 were published by the 5th New Zealand Computer Science Research Student Conference held in Hamilton, New Zealand, on 10–13 April, 2007 [49].

4.6 Next steps

The rest of this thesis illustrates the findings already framed in this chapter, using quotes taken directly from the interview transcriptions. The quotes are labeled with a combination of the speaker's team and the quote's associated key taken from the quotes database (see chapter 3), for example, [P1.108]. We also include the speaker's information: Their role on the team, and their team name, for example, Developer2, P2. Table 3.1 provides all roles interviewed and associates each participant with their team name.

As the ensuing discussion regarding the development activities of individual teams is rather fragmented, appendix A provides a more coherent and sequential description of teams P1 to P5's development activities.

Chapter 5

Before implementation begins

This chapter discusses our findings as they relate to the software development activities that took place before the implementation stage of the development effort began, i.e., up-front. Participants agreed that doing some interaction design before implementation began was essential in order to begin their development project. Activities that took place at the up-front stage, were concerned with obtaining information about the end users, modeling information about those end users and creating design artifacts based on that knowledge. A UI design specification was one of the design artifacts produced by the teams. Although all participants performed some interaction design up front, participants held contrasting opinions as to the amount of UI design that should be completed before implementation begins. These views influenced the teams' design strategy of their Interaction Design Approach, and is discussed in the next chapter.

The design artifacts played a role in estimating the work for the iteration and breaking up the development work into coherent elements that could be implemented iteratively. As these activities were performed up front, it became clear that the interaction designers established a solid direction for the product with input from their stakeholders and end users, and that the resulting product, based on the UI design created by the interaction designers, was expected to be usable and meet all the goals of

its end users. The information obtained and modeled up front was also found to drive the subsequent agile development activities.

In chapter 6 we examine the teams' Interaction Design Approaches in the context of the agile development activities.

5.1 Gaining a holistic view up front

On all projects in this study, an overall view of the scope of the project was known:

[P7.567] "We definitely knew what our problem was up front in this project. We definitely knew what we needed in the end to be able to claim victory in our project. We had a definite set of ideas and concepts around what this project was all about." — *Developer2, P7*

[P3.239] "You can not get around the finding out what it has to do aspect and then you can evolve your actual design off of that as you go." — *Developer1, P3*

Participants stated that gaining a holistic view up front clarified what needed to be implemented and guided development:

[P8.627,628] "It's good to be able to sit down and be able to have a whole, like a global view ...get a sense of, globally, what are we trying to achieve and coming up with a concept across the whole system." — *Project manager, P8*

[P3.294] "Up-front design gives you a clearer picture of what you're actually trying to implement and that makes it easier to make quick changes before you start implementing." — *Developer2, P3*

[P1.153] "It's hard to have a holistic view of the application when you haven't followed out all its framework: How am I gonna handle bread crumbs, how am I gonna handle, you know, main navigation, how am I gonna handle, like, user profile stuff ... So I created a really basic framework in the navigation model and then I just started working on tools." — *Product manager/UI designer*

[P1.186] “If it’s a big product, you can’t just design things piecemeal, like, XP implements things in little, small work units, you know, and you gotta have a holistic view, otherwise everything’s gonna be fragmented and you won’t have any cohesion. There won’t be any consistency or anything if you just kinda do it as you go.” — *Product manager/UI designer*

Participants agreed that some role in the team should have this holistic view in order to direct the overall UI of the product coming together:

[P6.479] “Facing the whole thing put together and going, ‘Ok, put that user interface together, put that user interface together, they’re both cool on their own but they don’t gel properly, they don’t work right together. One uses these words and the other one uses those words.’ ” — *Developer, P6*

5.2 Studying clients and users

Participants talked about their close collaboration with stakeholders such as clients and users of the product under development. By meeting with stakeholders during walk-throughs, surveys, workshops and site visits before development began, the interaction designers could elicit the system requirements from end users and gain a better understanding of their characteristics and work environments. The participant from P4 mentioned that requirements gathering focused on requirements that would affect the UI design:

[P3.212] “We use informal conversation with the customer, so someone tries to understand the domain that they’re working in, what the problems are and why they’re trying to achieve what they’re doing.” — *Developer1, P3*

[P4.336] “We are looking only for things [requirements] that would matter from the user interface design point of view. All the other

technical requirements and such have been collected previously.” —
Interaction designer, P4

Another team attempted to get input from as wide a range of stakeholders as possible:

[P2.16] “We’ll hold a workshop with the project sponsors, we’d have them with end users of the product, we’d have them with IT people, with the actuaries in the company, with the compliance people, as many as we can who will have input into the product or are using it in some form, or who are developing it in some form.” — *Interaction designer, P2*

Personas and scenarios, both well-known HCI techniques, were found useful by the teams for modeling the user’s goals in such a way that the team could construct UI designs that met these goals:

[P1.118,119] “I mean that was for the UI, but also for the user stories, one of the most important things to do. To really have these personas ... always knowing for who do we implement this. Who is the main user for this feature.” — *Engineering manager, P1*

[P2.66] “We design personas up front for our projects and we identify what their goals are, in using the product. We have to make sure that when we’re testing the product, we meet all their goals.” — *Interaction designer, P2*

[P2.18] “We write our user stories through what are called scenarios. Before they ever become user stories on a card for the development team, we do them via scenarios.” — *Interaction designer, P2*

Therefore, the product implemented based on the up-front design artifacts were expected to have a good level of usability built in:

[P2.70] “The usability will be built into the design of the screens, which the developers have.” — *Interaction designer, P2*

[P7.552] “We’re given these wireframes ...so we all think that what we’re doing is usable and good and pretty and things like that.” — *Developer1, P7*

Participants were convinced that doing up-front design and having interaction designers interact with stakeholders up front was the right thing to do, due to the positive feedback from their customers:

[P8.670] “My boss wasn’t fully convinced that it was the right thing to do to put UI people in front of the client and after about three months, the client walked in one day and ...I quickly mentioned this is the first project that we actually deal with an interaction designer up front and her comment was, ‘This is the only project where you’re doing this? Why aren’t you doing this on all of your projects?’ ...and she said this was the best one. She said, ‘I wouldn’t work in any other way and I don’t understand why you guys would work any other way.’ ” — *Project manager, P8*

[P2.29,74] “And it’s just what we believe in – we believe in agile development, we believe in interaction design and we combine the two together. So I can’t imagine us going away from those processes in the future ...it’s so intuitive how to use our software, it’s just very, very simple and straight-forward ...So that’s all because of the designers’ up-front work, because of up-front design and also because of the agile process we use. For me, personally, there is a huge difference in the satisfaction of our clients for this company than in previous companies I’ve worked for. And this is just phenomenal.” — *Interaction designer, P2*

The only exception to doing user studies was the team where the developers were users of their own product:

[P1.108] “Our customers are our product managers. I know for many teams where it’s the same, they just have the problem that their product manager wouldn’t use the application, so their product manager

kind of has to approximate, 'Ok, I think they will do this and this and this.' For us it's really not." — *Engineering manager, P1*

[P1.180] "The other thing that kinda helps is, and this is not a substitute for usability testing, but we're also active users of the product, 'cause we use it internally for our own development. It's a little easier for us to empathise with the users because it's like, you implement this feature, now I can't use my bug submissions the way I want to. So, it's a lot easier 'cause there's people internally, like QA, I can bounce ideas off. And they go 'You're gonna make my life hell if you do that.' " — *Product manager/UI designer, P1*

5.3 Designing for change

Participants found that creating some design artifacts — either in the form of personas and scenarios, high-level navigation maps or pen and paper prototypes representing the actual UI — were essential. Having the UI 100% complete was not necessary, as inevitable changes discovered during implementation should be accommodated in the design:

[P4.343] "What we currently try to do here at [Organisation], or what I try even sometimes to force through, is that UI design should be completed at least ninety five percent of the whole system before starting the implementation at all because otherwise it's simply not going to work." — *Interaction designer, P4*

[P1.111] "The UI designer actually can get away with not putting all the details and everything into it. Many things just work out during the iteration planning or during development ... he [the UI designer] doesn't have to make this absolute, final, ultimate thing that is then given to someone. You can get away with a seventy to eighty percent implementation." — *Engineering manager, P1*

A UI that was not completely 100% specified up front left room for interaction design decisions to be made during implementation. All partici-

pants in the study mentioned the fact that there were inevitable changes to the interaction design during development, such as implementation issues or issues that were not identified up front. Therefore, an incomplete UI design allowed adapting to newly discovered technology limitations, changes due to refactoring and changes due to newly discovered technological needs:

[P5.412] “[Change is] driven by the needs of the system or new things learned during the project and risks that didn’t get identified in the beginning.” — *Developer, P5*

[P4.368] “Although everyone says that the user interface is not in a straight link with the implementation, that’s not true. If there is a major refactoring of the user interface, it’s going to affect very much the implementation and it’s very costly.” — *Interaction designer, P4*

[P4.340] “It’s not realistic and not a good way of working to try to specify things to the nitty gritty detail, meaning that there will always be some kind of feedback from the developers when they find out that, ‘Hey, this is difficult to do,’ or ‘Have you thought of this kind of a situation, which came up now while trying to implement this?’ They give a seed for a need for redesign or completing the design, which is not sensible to do beforehand, because there are so many of these exceptional situations that the user interface designer would never guess, because he would need to know the internals of the system.” — *Interaction designer, P4*

5.4 Driving the development effort

With the exception of P3 and P5, the design artifacts created up front were found to drive the development of the product, both in terms of planning the development activities and developers having to match their implementation to the design artifact. Work estimates could be created, based on the combination of the user stories and the UI design, and the list of

features to be implemented in the iterations could also be created from the UI design:

[P1.177] “In the iteration planning meeting, we’ll hand the cards to the engineering manager and he’ll go through and get the estimates ... while he’s getting engineering estimates, I really have to think about it because they’re, you know, giving estimates and we’re making our plan based on this [UI design], and they don’t want to work 80 hours a week.” — *Product manager/UI designer, P1*

[P4.338] “So we’re kind of using the user interface design as the requirement for the developers ... and based on that user interface design it’s possible to break it into backlog items in Scrum and give estimates of how long this is going to take ... from my point of view, the user interface design quite heavily leads the release planning.” — *Interaction designer, P4*

Matching the implemented interaction design with the interaction design artifact was seen as the responsibility of the developers, which the interaction designers would check informally during the development iterations, or more formally during acceptance testing:

[P7.552] “Making sure it works would be one good thing, trying to make it match the wireframes as much as possible.” — *Developer1, P7*

[P1.171] “We’ve got a room – the observatory room — we’ve got two projectors side by side, we’ll pull up the html mock-up and then we’ll pull up the actual application page. We have this manual acceptance process, so if anything’s off, it’s unaccepted and it goes back to the engineer.” — *Product manager/UI designer, P1*

As a result, the design artifacts played a role in the communication between the interaction designers and developers throughout development, as features of the design were discussed and clarified for implementation:

5.5. ADVANTAGES OF INTERACTION DESIGN BEFORE DEVELOPMENT BEGINS81

[P4.371] “We currently have the PowerPoint for the HTML prototype, and what we do is we give them [developers] a printout of that so that we can discuss and communicate with those prints. They [developers] can circle things on that piece of paper around ‘What does this do?’ and write comments and so on.” — *Interaction designer, P4*

[P7.538] “These are just still wireframes. We talk about, ‘What happens when you click here?’ and then we have discussions ... They don’t show workflows and a lot of it relies on us to communicate with the people who have made those wireframes or have the expertise.” — *Developer1, P7*

5.5 Advantages of interaction design before development begins

The participants were clear about the advantages of doing interaction design before implementation began. Among the teams three advantages directly attributed to up-front design before implementation begins, were identified:

1. Saves cost and time
2. Increases user satisfaction
3. Increases usability testing productivity

Cost and time For teams P4 and P5, up-front design was seen to contribute to cost and time savings by ensuring better project estimation and prioritisation:

[P4.369] “Based on the user interface design, you can make very specific estimates of how much work is it going to take to implement this ... it’s possible to make sophisticated judgments about what we should do first, how much this is going to take, what creates added

value for the user, which things should we tackle first, give very good and exact work estimates.” — *Interaction designer, P4*

Participants also noted that up-front design helped designers come up with the best possible design, while keeping to the customer’s budget:

[P4.369] “If you do an up-front user interface design you can communicate to the customer how much it’s going to cost, how much it’s going to take to build the whole system, so the customer can again tell back that ‘Ok, I don’t have so much money,’ so we can try and prioritise and say that ok, with this much money, we think that the best bang for buck is to cut it over here that is either underneath the budget or just a little bit above it, so that they can then try to adjust it.” — *Interaction designer, P4*

Negotiations with the customers before development begins prevents this from becoming a potential bottleneck during implementation. Participants explained how negotiations with customers about the interaction design during the implementation stage, could hold up the development work:

[P4.338,347] “We make a deal with the customers, ‘This is the amount that we think this is going to take,’ and then once they say ‘Go,’ we start implementing that ... We try to actually push this so that this would have been done before hand, so that the features that the developers were working with, the work would have not ended and we would not have needed to put this project on hold, but our situation was such that our customers did not believe that this was going to happen.” — *Interaction designer, P4*

[P5.426] “We had just one person responsible for creating the HTML prototype ... if you consider the responsibility of making these changes it’s kind of within this one person and he has to have this tight interaction with the customer, then that interaction might become a little of a bottleneck.” — *Developer, P5*

5.5. ADVANTAGES OF INTERACTION DESIGN BEFORE DEVELOPMENT BEGINS83

User satisfaction In teams P1, P2 and P6 up-front design was seen to have a positive impact on the final product's user satisfaction and consistency:

[P2.73] "Well, I think from my experience, when I worked in companies other than this company, I was involved in user interface, and we weren't putting in nearly as much effort into the [up-front] design of the software and I have to say that the user satisfaction, our scores were very low. Not only that, but there was weeks and weeks and weeks of training on the systems in order to be able to understand how to use them." — *Interaction designer, P2*

[P1.155,187] "Just create some up front consistency, like, what do buttons look like, where are they placed, what do tables look like, how do users interact with tables, what do forms look like, how do you get from a table to a form and then back to the table, like, basic interaction models. So, kinda like a style guide ... So, having that up-front consistency in designing the behaviour into it." — *Product manager/UI designer, P1*

This was especially true when teams could demonstrate some mock-up for requirements verification before implementation:

[P6.475] "I very quickly mock up something that usually has a scripted set of behaviours ... so that way our systems engineering guys who'd gather the requirements can actually pick up a real application and play with it and see, 'Ok, well, do I like the layout? Do I like the size of it, how it looks on the screen? Do I like how it's displaying the information? Do I like how that transition works?' that sort of thing ... And then they usually say 'I like this, or I don't.' " — *Developer, P6*

Usability testing productivity Team P4 credited up-front design with increasing the productivity of usability testing:

[P4.386] "That [usability testing the final product] would find out the small stones in your shoe, because we believe that big stones in front

of your shoe have been removed mostly by the user interface designer doing design up front and iterating and evaluating the design with the users.” — *Interaction designer, P4*

5.5.1 Another notion of up front

Creating an interaction design during the stage before implementation begins is one notion of creating a design up front. Another is the case where interaction designers do interaction design before it is implemented in the next iteration. In this case, participants identified a further two advantages to creating an interaction design before the developers implement it as functional software in subsequent iterations. These are advantages as a result of the interaction design being done before that interaction design is implemented, rather than completing all interaction design before implementation begins:

- Mitigates the risks of the developers’ work
- Anchors the UI design

De-risking development work The interaction design work performed up front by team P5 and P8 was seen to de-risk the work of the developers later in the development effort:

[P8.658] “I think, the more forward thinking they [interaction designers] do, the more they de-risk what’s happening now because they’re looking at it from, like a little further away and they’re saying, ‘This is coming down the line..’ ” — *Project manager, P8*

[P5.458] “It can take a while to work through all the pages and while developing, we may run into problems two to three weeks into the development iteration. This is a problem schedule-wise, as the later the changes propagate into the schedule, the bigger the impact they have on the overall development effort.” — *Developer, P5*

Anchoring UI design With concrete design artifacts such as pen and paper prototypes and wireframes, P7 found that the UI design can be anchored. This provided a stable guide for the UI implementation, as seen in section 5.4. Design artifacts also helped to reduce the probability that one group changed the UI design without communicating with the other group:

[P7.560] “They [interaction designers] change things ... Now that we have the wireframes ... Not as frequently anymore. At the beginning, it was really hard.” — *Developer1, P7*

There were also benefits for the communication between designers and developers, as artifacts helped to ground their understanding:

[P7.561] “I talk about these wireframes like they’re the word of god – but it’s nice ‘cause I have something to reference, right. Before we had nothing and we’d be like, ‘Is this what you mean?’ ‘No.’ ” — *Developer, P7*

5.6 Summary

This chapter discussed our findings as they relate to the software development activities that took place up-front in the development effort. There was agreement among the teams that the period of time before implementation began was the most suitable for activities related to understanding the end user and obtaining an overall view of the project under development. These were seen to be essential activities to begin the development effort. Participants believed that the holistic view gained during up-front design helped clarify ideas on what had to be implemented. Up-front UI design was considered beneficial when it was allowed to change during the implementation stage and often ended up driving the rest of the development activities. Finally, we discussed the advantages that were experienced by participants as a consequence of doing interaction design

up-front. Further discussion of the Interaction Design Approaches is continued in chapter 6.

Chapter 6

Interaction design approaches

The previous chapter explained that the teams in our study considered some up-front interaction design work essential to start the development effort. This chapter continues the discussion, with a focus on the way the teams organised their interaction design and agile development activities. We explain the teams' Interaction Design Approaches, i.e., how they approached their interaction design in terms of its design and its implementation. Within the data, distinct patterns of interaction design emerged. These patterns were essentially strategies the teams employed in order to deal with the design aspect of the interaction design and its implementation.

The combination of a team's *design strategy* with their *implementation strategy* is what we refer to as the Interaction Design Approach, as outlined in chapter 4. For the design aspect, two strategies emerged, while three strategies emerged for how teams dealt with the implementation of the interaction design. In the following sections we discuss the combinations that were observed in our data, introduced in chapter 4, focusing first on the activities that defined the design strategies and then on how the interaction design activities were combined with the planning, implementation and evaluation activities of the agile development iterations — defining the implementation strategies. In chapter 7 we discuss how in-

teraction design and agile development were seen to work together for considerable advantage to the overall development effort.

6.1 Overview of interaction design approaches

The first Interaction Design Approach, discussed in section 6.2, combines Comprehensive Design with Refinement. These teams understood interaction design to occur before the agile development process begins, and therefore, created a comprehensive interaction design before implementation began. They designed their UI such that it was a representation of the entire system under development, which was then implemented during the agile development iterations.

The second approach, discussed in section 6.3, combines Evolutionary Design with Looking Ahead. These teams merged the interaction design and agile development processes so that the interaction design activities were interleaved with the agile development activities throughout the development effort. Participants who subscribed to this view produced a UI design that only implemented the features from previous iterations and the features that had been selected for a set number of iterations ahead. In this way, the UI design was allowed to evolve during the implementation stage.

The third and fourth approaches are combinations of each of Comprehensive Design and Evolutionary Design with Parallelisation. These are discussed in section 6.4. Their UI design and evaluation activities and the implementation activities were performed in parallel. As the developers were implementing the system, the UI was iteratively designed and evaluated separately.

As table 6.1 shows, the team's agile development methodology appeared to have no identifiable relationship with their Interaction Design Approach.

Team	Agile	Design Strategy	Implementation Strategy
P1	XP	Evolutionary Design	Looking ahead
P2	XP	Comprehensive Design	Refinement
P3	XP	Evolutionary Design	Parallelisation
P4	Scrum	Comprehensive Design	Refinement
P5	Scrum	Comprehensive Design	Parallelisation
P6	XP	Comprehensive Design	Refinement
P7	Scrum	Evolutionary Design	Looking ahead
P8	XP	Evolutionary Design	Looking ahead
P9	XP	Comprehensive Design	Refinement

Table 6.1: Teams and their Interaction Design Approach

6.2 Comprehensive Design

P2, P4, P5, P6 and P9 were teams that performed Comprehensive Design:

[P2.3,5] “The way it generally works from the top is our domain guy/customer will have a vision. Marketing will go off and verify the vision, basically to see if it’s viable, identify the market gap, or whatever. It will then be passed down to the interaction designers who will then come up with a design for an actual product that we’re going to develop. And that’s when it starts sinking down to our level. We’ll start having planning meetings and stuff like that. Come up with the actual schedule for putting the project together. That’s when XP really takes off.” — *Project Manager, P2*

[P9.700] “Lots of mock ups, and story-boards for what the user experience would look like [designed before development begins].” — *Developer, P9*

Quote P2.3,5 above indicates a preference to have the UI designed completely before implementation began and then handed off to the developers to implement. The number of features included in the UI design, i.e.,

the coverage of features in the UI design, included the entire list of features envisioned by the team at the outset of the project:

[P2.23] “Before it gets into development, the user interface is more or less, 90% defined. So there wouldn’t be that many changes once it goes into the development iterations.” — *Interaction designer, P2*

The participant from P4 explicitly talked about removing UI design from the agile development iterations:

[P4.337] “What we’re trying to do is actually get the user interface design out of the iteration cycle, so that we would have specified the functionality and how the user interface exactly works and the iterative part would be actually implementing that.” — *Interaction designer, P4*

This hints at a view of interaction design and agile development as two separate processes, belonging to distinct domains. The Interaction designer on team P2 communicated the sentiment that practitioners are not encouraged to contribute outside of their domain:

[P2.31,56] We as designers have great respect for what the developers do and the developers have great respect for us. Even with great communication going, we’d never assume to make a suggestion about something we don’t know very much about . . . I can’t imagine why a developer would be designing screens because their training lies in a whole different area to me, and their skill set lies in a different area so I just don’t understand how companies can do that.” — *Interaction designer, P2*

An important insight was that interaction design performed before implementation began was seen by participants to be different to up-front code design. For example, up-front code design was seen by participants to produce costly waste, whereas being able to refine the UI design with the customer up front to make sure it is correct, was considered essential:

[P2.57] “The kind of issues that you get in up-front code design are not the issues you get in up-front UI design at all. Code design comes out of just the amount of waste you get from people doing two, three jobs. Do the same job two or three times in different forms. You might have something described on paper, then you might have something described in boxes and arrow diagrams, then you might eventually start hand coding it then you’d probably have to fix it later because you don’t have any tests, that kind of thing. There’s an awful lot of waste involved in that, which is the main thing XP was trying to fix in the first place, but with regards to up front interaction design, you know, putting together screens, in Photoshop or whatever and iteratively running it by customers and things, to make sure that the design itself is correct. So it’s a whole different domain, basically.” — *Developer, P2*

[P4.400] “We’re not writing a specification that is not true, we’re trying out the system in the cheapest possible way in a very agile fashion but we’re doing it with pen and paper and it’s to build a system. This pen and paper thing is not design up front, it’s defining what the system is in a faster and a cheaper way. If it would be as cheap to implement the system at the same time, well then go for it. But if I can draw in five minutes so much user interface that it takes two months to implement, I really do not think that that is the best way to tackle things.” — *Interaction designer, P4*

Participants agreed that up-front interaction design does not clash with the agile principles which discourage up-front design, as long as it is iteratively refined with feedback from evaluation:

[P2.14] “So ultimately what we do is we build a certain amount of our product using our product visionary, and then we go in and refine the rest with our clients.” — *Interaction designer, P2*

[P2.53,57] “Personally, I don’t think XP warns about doing up-front design. I think it warns about doing a huge amount of code design up front. I think there’s a difference.” — *Developer, P2*

[P3.238] “XP says nothing about finding out what you’ve got to do up front. Just make sure you put it through a feedback cycle.” — *Developer1, P3*

[P5.439] “And then at that point, when it [UI design] is passed to the person who does the HTML prototype ... it’s an iterative process with the customer because he has to come up with a consistent look and feel.” — *Developer, P5*

Once the UI design had been established in the period before development began, the UI design was not expected to undergo any major redesign during implementation, as the interaction designers created the best design up front. The anticipated changes in the UI design were expected to be minor issues regarding implementing the design utilising the technology with which the product was being developed:

[P2.24] “Very little change [during implementation] really because we put so much effort into the design.” — *Interaction designer, P2*

[P4.397] “If there’s a good design for that, there’s a reason why the design is good and there is no reason for the implementer to create something else, if what’s a good design is buildable.” — *Interaction designer, P4*

Of the Comprehensive Design teams, only P2 mentioned testing with users once implementation began:

[P2.76] “We try and do user testing a good bit before we were having our final release” — *Interaction designer, P2*

P4, and P6 did not perform formal usability evaluations with users, due to a lack of customer interest in usability testing:

[P4.378] “You have to have in the budget usability tests and so on. Currently that’s not realistic with most of these pieces of software. And the customers themselves are not even very interested in that.” — *Interaction designer, P4*

[P6.511] “Not intentionally not interested, they’re not focused in that direction. So yes they’re not interested, but it’s not a conscious omission.” — *Developer, P6*

For P2 and P4, users were involved in UI design evaluations which occurred up-front:

[P4.350,352] “We create a design, test it ourselves, find that this is simply, ‘We think that this would work,’ and then we try to evaluate it with users and when we find out the problems, then we do redesign and then evaluate again . . . User testing takes place up front, because it has turned out that that kind of iterating with built code is most expensive.” — *Interaction designer, P4*

The participant from P4 clearly indicated that evaluating with users before implementation begins removes the major usability flaws in the UI design:

[P4.386] “Usability testing the final product would be very reasonable because that would find out the small stones in your shoe, because we believe that big stones in front of your shoe have been removed mostly by the user interface designer doing design up front and iterating and evaluating the design with the users.” — *Interaction designer, P4*

6.2.1 Implementation as Refinement

As the interaction design was either created by skilled interaction designers, or refined with end users before implementation began (or both), P2, P4, P6 and P9 considered the implementation of their Comprehensive Design as a process of refinement. Figure 6.1 shows the structure of the agile iteration in the Refinement strategy.

Planning

Planning, in terms of work estimates and prioritisation of features for implementation in the iterations, were based on the interaction design. After

the overall interaction design had been created, the work of implementing the design was broken up into components that could be implemented iteratively:

[P4.337,356] “Based on that user interface design it’s possible to break it into backlog items in Scrum and give estimates of how long this is going to take, and then we make a deal with the customers ...the user interface design quite heavily leads the release planning.” — *Interaction designer, P4*

[P6.471,480] “So there is a specifications document for the big stuff, and it’s vague enough that we can refine it as we go along, via stories. So we can actively transform that into stories in our queue ... They want to write big documents so we let them write big documents and then we fracture them.” — *Developer, P6*

Implementation

The developers matched their implementation of the interaction design with the UI design artifact produced by the interaction designers up front. The work of implementing the UI was not distinguished from other system implementation work:

[P4.337] “So we’re kind of using the user interface design as the requirement for the developers.” — *Interaction designer, P4*

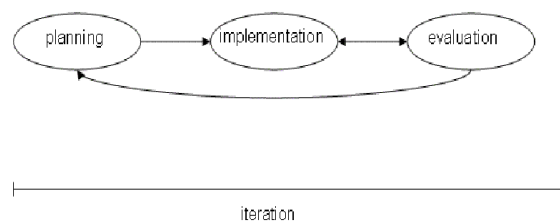


Figure 6.1: Refinement Strategy

[P9.700] “The UI mock-ups are completed by the data-visualisation expert and lead developer, before implementation begins and the developers implement the mock-ups (in terms of what the UI should look like) and the functional document in terms of how the UI should behave.” — *Developer, P9*

[P6.480] “We wouldn’t implement the UI and then later implement the back end. It would just be all put together.” — *Developer, P6*

At this stage of the iteration, coding was the main activity. The UI design work consisted of small adjustments to the interaction design to allow for it to be implemented, but was considered a small percentage compared to the UI design performed up front:

[P4.340] “There will always be some kind of feedback from the developers when they find out that, ‘Hey, this is difficult to do,’ or ‘Have you thought of this kind of a situation, which came up now while trying to implement this?’ They give a seed for a need for redesign or completing the design . . . So the five percent that these developers do — they do not do the design — they give ideas of where more design is needed.” — *Interaction designer, P4*

P2’s interaction designers performed their own interaction design evaluations of the implemented interaction design on a daily basis:

[P2.69] “The designers test it on a day to day basis, give feedback back to the development team to ensure if anything was missing, that we’d write a card for it and it will be captured.” — *Interaction designer, P2*

Interaction designers also performed informal evaluations by checking the developers’ implementation of the interaction design as they worked, to ensure the implementation matched the interaction design specification:

[P4.384] “We also do this unofficially that they [developers] say, ‘Well, look, come and check this out,’ so we go and check it during the implementation.” — *Interaction designer, P4*

Evaluation

Evaluation of the implemented interaction design occurred at the same time as evaluation of the whole product. Acceptance testing was thought to be the most convenient time to do this:

[P2.80] “You only really need to unit test the system that’s building your UI. So the data that describes what the UI is going to look like doesn’t need to get tested in anything other than acceptance tests.”
— *Developer, P2*

Evaluation of the usability of the product could consist of more formal testing with customers, if the project allowed the time for it, but this was not scheduled as part of the agile iteration:

[P4.385] “One thing we could do, but we probably won’t, since our customers haven’t been very interested in such things, would be to usability test the final product.” — *Interaction designer, P4*

Large interaction design fixes resulting from the evaluations were written as new user stories on user story cards (carded) and could be incorporated into future iterations:

[P2.75] “We mainly card them and negotiate between the developers, ourselves and the domain expert when we put those fixes in, which iteration those fixes will go into.” — *Interaction designer, P2*

6.2.2 Key conditions

From the above discussion we infer that inherent in the Interaction Design Approach that combines Comprehensive Design with Refinement are two key conditions that allow teams to develop software using these strategies:

- Interaction design and agile development are different processes that should remain separate, and

- Changes to the interaction design once implementation has begun will be minimal.

Participants saw a clear boundary between the interaction design process and their agile development process. Approximately 90% of the interaction design was seen to occur up front, followed by the agile development process, which iteratively implemented the UI design. User involvement also occurred largely up front, as the UI was iteratively designed and redesigned with user input. Due to the view of ‘separateness’, interaction designers and developers were expected to work within their own domains and input from the developers was expected to be minimal — mostly drawn from a technical domain outside of interaction design. As a result, the role of the developers with respect to interaction design was seen as that of implementer — their implementations being driven by the interaction designs produced by the interaction design ‘experts’.

The development iteration cycled through planning, implementation and evaluation activities, but there was little or no interaction design work being done. The anticipated changes to the UI design during implementation, were minor design changes required to implement the design with the technology being used to develop the product. Design during the agile iteration was only required when these technical impossibilities were encountered by the developers.

6.3 Evolutionary Design

P1, P3, P7 and P8 were teams that performed Evolutionary Design, where the complete UI design was not seen as being ‘handed over’ to the developers to implement, and was never talked about in this way:

[P1.111] “It’s not that the UI designer has to come up with the full UI design, everything final, all Java Scripts and everything in it. Then goes to the engineers and says ok, implement this and then sees it

four months later and has to, like, look through it, how does it work?"

— *Engineering manager, P1*

[P7.531] "If people are gonna ask you what your product is going to be in a year's time you can't really tell them. Our design is like a living breathing thing, it changes all the time." — *Developer1, P7*

Instead, the UI was designed in increments, incorporating only features in the UI design that could be implemented in a single iteration:

[P3.231,234] "We sort of have a longer term plan of what we want to achieve, and then every week we recap that and look at what we're going to put into this week as the work for the week, so we discuss any bugs or issues that are out there and put those into the work ... The idea is to just try and keep the software in a workable state all the time." — *Developer1, P3*

[P7.571] "The first week of every sort of sprint, we'd ask, 'What do we need to do in terms of design to further our understanding of what needs to be built.'" — *Developer2, P7*

[P1.99,111,129] "'Cause the UI is designed, at least until the next iteration. Sometimes we don't have the UI for the full next release, just for the next iteration ... these are all the user stories that we need or that we are planning to implement, so I make a UI for all these fifteen user stories." — *Engineering manager, P1*

Since the interaction design at each iteration was not necessarily what the interaction design would eventually be, participants referred to an overall idea or metaphor that guided the evolution of the interaction design toward a coherent implementation:

[P1.129] "As long as you have your overall, kind of, goal in mind, like this was the design that we thought for this whole feature, so kind of the UI metaphor in this case. It's something that you don't write down, it's not written down, the screen has to look like this and so on. It's kind of an idea, ok, ultimately, it should look like this or it should behave like this." — *Engineering manager, P1*

Participants included the experience of the team and existing design standards as mechanisms that guided evolution of the interaction design:

[P1.130] “In the end we have things like experience from previous projects, we have things like design factors. Every now and then you just know, ok, this is the direction I should take.” — *Engineering manager, P1*

[P3.295,325] “At least you can see potential problems [with the interaction design] based on experience, standards and conventions ... I think that you can get it right based on experience ... a lot of patterns and standards and conventions exist. You’re not always inventing something new.” — *Developer2, P3*

The Evolutionary Design strategy ensured that only the features that delivered business value at that particular time were implemented in the interaction design. Further, Evolutionary Design allowed feedback from evaluation activities to influence and change the interaction design:

[P1.204] “And with XP it’s like, these are my cards, this is what I need to design for. I don’t care about the next release. I’m not even thinking about that. And nine times out of ten for us, if we did try to think about the next release then when we designed it for that then we didn’t end up implementing those features anyway.” — *Product manager/UI designer, P1*

[P3.239] “With the user interface you gather as much information as you need to do some kind of real thing and then you put it through an iteration ... back into real code and then you’ve got something that people can play with and look at and then you can go through another iteration or another cycle of ‘Is that any good? What should we be thinking about there?’ ” — *Developer1, P3*

[P3.291] “We have parts of the user interface we know needs improvement, and a lot of that is based on the feedback that we get, so then we will schedule it in.” — *Developer2, P3*

[P7.576] “Every couple of months those user feedback sessions go on and they are used as input into our media people, into our user experience people and then they feed back into our design.” — *Developer2, P7*

Releasing real working software into its context of use in order, to at least deliver some functionality was seen as acceptable. User feedback would then be received and acted on once the software had been in use:

[P3.256] “Some stuff we’ve done we’ve gone, Well, we’re so distant from these particular users, we think we understand the problem, we’ve created the user interface, let’s just release it, and it might be difficult for them to use, but we can get back to them and ask them how they found it and get feedback that way, by releasing the software, effectively. At the minimum they get the functionality. It gives us another feedback loop that we can go to those users and say ‘Does it work for you or doesn’t it work for you?’ ” — *Developer1, P3*

Whereas teams following the Comprehensive Design strategy considered up-front user involvement as an opportunity for evaluating the complete interaction design, the Evolutionary Design teams required just enough user involvement in order to obtain interaction design information to start the implementation, since the interaction design evolved during implementation:

[P3.239] “With the user interface you gather as much information as you need to do some kind of real thing and then you put it through an iteration . . . There shouldn’t be any up front design of the underlying code architecture.” — *Developer, P3*

For Evolutionary Design teams, evaluating with users took place during implementation, using working software. The implemented interaction design was expected to be improved on with user or customer feedback:

[P3.259] “It’s not the best user interface but it delivers a piece of functionality for now and we can work on improving it later.” — *Developer1, P3*

[P8.644] “After the user acceptance testing, we find that it often goes back to the developers or the interaction designers because of, ‘Yeah, we don’t like the way it was implemented.’ ” — *Project manager, P8*

A unique feature of one of the XP teams in this study, P1, was the view that there must exist some way to apply the XP practices to UI design. P1 expected that the ability to do this would improve the overall development effort:

[P1.202] “If you can take the XP process and somehow apply that to the user interface design process, I think that will improve things vastly.” — *Product manager/UI designer, P1*

Although it was not clearly understood *how* the XP practices should be applied to interaction design:

[P1.114] “I think that the paradigm that you have in XP where you say, implement it as simply as possible and refactor it and through the refactoring you generate a good framework. I think that paradigm works fantastic on the back end, works fantastic with integrated systems. I think on the front-end, then again maybe I’m wrong, maybe I just haven’t seen it yet implemented properly, but I think on the front end you have a much, much harder time.” — *Engineering manager, P1*

For team P1 the ideal combination of interaction design activities with agile development activities was one where all the agile development activities that apply to the coding part of development are also applied to the interaction design part. This differs to other Evolutionary Design teams (P3, P7 and P8) who agreed that the interaction design activities should be interleaved with the agile development activities. Despite P1’s different view, in practice, P1’s interaction design and agile development activities were organised in a similar way to P7 and P8.

6.3.1 Implementation as Looking Ahead

Three of the nine teams in our study developed their product according to the Looking Ahead strategy when coordinating their interaction design and agile development iterations: P1, P7 and P8. The up-front strategy of these teams was characterised by Evolutionary Design as described above. Figure 6.2 shows the structure of the agile iteration in the Looking Ahead strategy. As the participant from P8 explains, the interaction designer designed the interaction for a fixed number of iterations ahead of what the developers were implementing in an iteration:

[P8.643] “UI is working on the third iteration, so they’re always looking ahead at what’s coming, and I’ll even say three, four, while the developers are actually really focusing on [second iteration].” — *Project manager, p8*

[P7.530] “So our requirements are fixed for what we want to have achieved at the end of the sprint, but aren’t fixed on the sprint to sprint basis.” — *Developer1, P7*

Planning

Planning the work for the iteration was based on the prioritised features for the iteration:

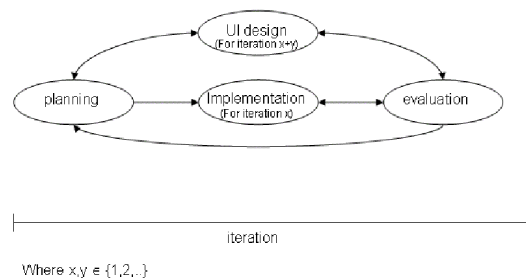


Figure 6.2: Looking Ahead Strategy

[P1.117] “In release planning when we make our high level estimates, usually it doesn’t have too much to do with the UI.” — *Engineering manager, P1*

[P1.160,177,178] “When it actually comes into the actual iteration planning then we’ll go through it [UI design] again. We get more granular point estimates and there I’ll also take feedback ...if there’s something that needs to be changed, or someone has a better idea, it’s like, it doesn’t change everything ...they’re, you know, giving estimates and we’re making our plan based on this...so we’ll say, we can do 14 points in an iteration, you know, we got 20 points, OK, what are we gonna do. So we’d leave these out and these two need to be rethought about a little bit anyway, so we’ll do most of them and we’ll put these in a little pile and then we’ll put these in the next time and then we’ll continue.” — *Product Manager/UI Designer, P1*

[P7.529] “We have a solution spec, which has all the use cases and things like that in place. It’s supposed to be our overall product because our overall product will be able to achieve all of those things. But when we try to break down into our thirty day chunks we look at one scenario in particular, not necessarily out of the solution spec, and then we pull out the requirements from that scenario. ‘Ok, so what do we have now, this is the scenario we want to work towards, what’s missing, what’s the gap?’ And then we’ll take those items and plan with those. And if we can contain them all that’s great and if we can’t, we modify the scenario until it’s something we can achieve in the thirty days.” — *Developer1, P7*

[P7.532] “When we’re doing our planning meetings, we look at the items that we have on the backlog for the current sprint and we ask ourselves, ‘Ok what do the users need to be designed?’ ” — *Developer1, P7*

Interaction design changes, based on stakeholder feedback, could also be prioritised and incorporated into the work for the iteration:

[P7.576,586] “Because again, every 30 days we have a planning session that includes those people as well, so they are allowed the opportunity to put their criticism on the table and say, ‘This is what I think needs to be done, to be changed.’ ... And after that we look at all the listed items we have on the backlog we have, and roll in all the feedback we’ve got from that demonstration and feedback session. And then we plan, based on that, what the next one should contain. What the highest priority items are, what makes the most amount of sense to implement.” —*Developer2, P7*

Implementation

The developers matched their implementation with that of the interaction design for that iteration:

[P8.638] “The developers, when they are assigned a user story, for example, user has to be able to search by using the corporation number or something like that, they’ll go to the interaction designer, talk to the interaction designer about what are the mock-ups, what does each field mean and each label and stuff like that, and also what are the user acceptance tests around that.” — *Project manager, P8*

During the iteration once the developers had completed a user story, they approached the interaction designers — either ad hoc or at a daily meeting — so the interaction designers could verify that the implementation matched the UI design specification:

[P8.648] “Throughout the iteration there’s little spots where every time that a developer finishes a user story, then there’s a bit of testing of the UI ... the interaction designer provided the requirements to the developer, the developer says, ‘I’m done,’ so the interaction designer says, ‘Well, show me that you’ve done what I’ve asked you.’ ” — *Project manager, P8*

[P7.583] “We have a daily meeting with those stakeholders [interaction designers] as well, saying ‘I have an issue: what you gave me,

I just can't implement,' and that sort of feedback happens almost daily." — *Developer2, P7*

The implementation activities were similar to teams employing the Refinement strategy, however, as quotation P8.643 above mentioned, there were also interaction design activities being performed at the same time. As the implementation activities were taking place, the UI was being designed for future iterations.

Evaluation

As with Refinement, the interaction design was evaluated at the end of the iteration during XP's acceptance testing or Scrum's demonstration session:

[P1.101] "There [during acceptance testing] we walk through and there we actually don't mind giving a feature back to engineers, saying that it has to be implemented again, or it has to be changed because of usability. Sometimes the feature actually works, it's there but then we see it there and we're like no. It may be the order, it might be the screen flow, the navigation or something and we're like, no it really doesn't work well." — *Engineering manager, P1*

[P7.526] "Every thirty days we have something that we demonstrate and we can play with and we can show and it's working code. I think that's really neat, and it's tested." — *Developer1, P7*

[P8.649] "Towards the end of the iteration, I'd say the last couple of days, and the beginning of, the days of the next iteration, the interaction designers are testing the amalgamation, the combination, the integration of all of these other user stories" — *Project manager, P8*

Evaluation with the client was a regular occurrence for P7 and P8 and was scheduled for the end of every iteration. The client was evaluating the working software the developers had implemented in the last iteration:

[P7.586] "We have a demo to all the stakeholders inside and outside our organisation, whoever they may be, for that particular month's worth of work." — *Developer2, P7*

[P8.650] “The client is really testing the UI for usability and they’ll often tell you they’re not testing the UI, they’re testing the application, but really if there’s something wrong with the UI they will tell you, ‘The color’s not right, the label’s not ok.’ There’s testing with users at the end of every iteration.” — *Project manager, P8*

All evaluation of the interaction design was a manual process:

[P1.100] “I think the closest that we have to a usability testing is that we accept all the features manually, so we have to just stay in the acceptance meeting then we walk through the feature manually, see if it works.” — *Engineering manager, P1*

[P7.543] “UI testing happens with the other testing that goes on. We haven’t automated it yet. Most of our testing of the runtimes are automated. We don’t have a tool or we haven’t used a tool yet to automate our UI testing.” — *Developer1, P7*

Large interaction design fixes resulting from the evaluations were written as new user stories and could be incorporated into future iterations, whereas smaller adjustments were fixed right away — similar to the Refinement teams:

[P1.102] And then usually we make a decision on the fly, or, yeah, we just give it back [to the engineers] and we change it, or we make it another user story. And it’s a very ad hoc decision.

[P7.545] “Most of the fixes [resulting from UI testing] are things not working, so we generally are able to fix them before the end of the sprint. The fixes such as, I don’t know, we need to add search functionality or things like that would be almost backlog items, so they’d be added to the backlog and then revisited when we’re planning for the next sprint, prioritised and stuff.” — *Developer1, P7*

Issues other than the visible and interactive aspects of the interaction design would be identified and improved during implementation:

[P7.589] “Testing, in terms of functional capabilities, performance, the more underlying type things, that happens within the month as the product’s being built. So our implementers will see a performance problem or something, “This table just gets too big I can’t even build a scroll bar that you can go up and down in, cause there’s over a million entries.’ They’ll raise fundamental concepts like that as they see them.” — *Developer2, P7*

6.3.2 Key conditions

From the above discussion we infer that inherent in the Interaction Design Approach that combines Evolutionary Design with Looking Ahead are two key conditions that allow teams to develop software using these strategies:

- Interaction design and agile development can occur simultaneously, and
- User/client feedback drives the interaction design work and subsequent software implementation.

Interaction design and implementation iterations were occurring simultaneously and the interaction designers did not ‘hand off’ a complete interaction design to the developers. Instead, the interaction design evolved along with the implementation, in a way that the interaction design was not only iteratively implemented, but also iteratively designed iteration by iteration.

The interaction design included features for one or two iterations ahead of the design being implemented. When the development iteration was at the implementation stage, the interaction design iteration was at the design stage, and the interaction designers were designing for future development iterations. Therefore, the developers were only implementing the design that the interaction designers had created for that iteration, adding the features that had been selected for implementation in that iteration,

to the interaction design. Proceeding in this way allowed feedback from user evaluations scheduled for the end of the implementation iterations to influence the subsequent interaction design and implementation.

6.4 Parallelisation

P3 and P5 followed a somewhat different strategy to the other teams in our study. Their UI design and evaluation activities and the implementation activities were performed in parallel. As the developers were implementing the system, the UI was iteratively designed and evaluated separately:

[P5.426] “We’ve driven the project so that the customer develops the idea and the user interface designs by themselves, and not leaking any information out of it before they get it ready, and then after that we usually end up in a bit of a hurry in developing the HTML prototype, and also developing the system in parallel.” — *Developer, P5*

[P3.289] “We’d first make a few design sketches. We quite like to start with either pen and paper or PowerPoint. That can go on at the same time as doing the coding.” — *Developer2, P3*

As can be seen from P5’s quote above, this team had a comprehensive design handed to them by another team to implement, however, P5 also developed their own HTML prototype. This prototype was created in parallel with the system implementation. P5 was the only team to combine a Comprehensive Design strategy with a parallelisation implementation strategy.

P3 followed a Parallelisation implementation strategy and evolved the UI along with the system — in a similar way to teams who combined Evolutionary Design with the Looking Ahead strategy. The difference with combining Evolutionary Design with Parallelisation was that the interaction design iterations were not required to keep up with the development iterations and the interaction design did not necessarily undergo further design at every iteration:

[P3.290] “Yeah, we can do several coding iterations and not worry about the UI and can come back to it. After we’ve done sketches and we’ve designed something and we’ve had the latest review, it can just sit there while we’re working on the code and we might go back to it.” — *Developer2, P3*

[P3.328] “With faster and quicker iterations in agile maybe ... With the user interface it takes longer to get feedback, so it doesn’t always line up.” — *Developer2, P3*

One of the participants from P3 pointed out the importance of maintaining separation between the UI design and the system code:

[P3.289] “We want to write really good code, so it’s really easy to switch the user interface. It’s not too locked to the code, so we keep different layers. We like to keep the code separate from the user interface so if it changes ideally you can just plug in another one.” — *Developer2, P3*

Figure 6.3 shows the structure of the agile iteration in the Parallelisation strategy.

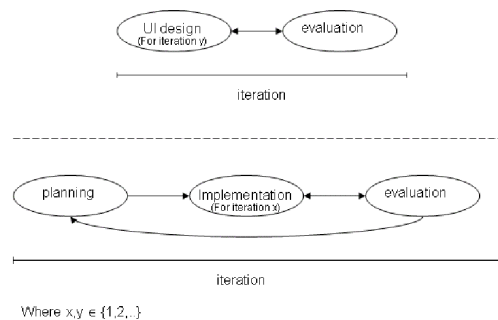


Figure 6.3: Parallelisation Strategy

Planning

Planning in the Parallelisation strategy is noticeably different to the Refinement and Looking Ahead strategies, in that the UI design had little or no role during the planning activities:

[P3.235] “The interaction design is not explicitly planned for; it’s just done as part of the iteration.”

[P5.411] “No we didn’t use user stories ... we drove down to the actual task level with the planning of each iteration ... and reformulated concrete tasks that would be easy to manage within iterations and easy for the customer to accept and test.” — *Developer, P5*

[P5.427] “The work phase of creating the prototype is outside of the iteration planning because the work is not cut out in tasks, which reflect actual requirements, they’re just like getting pages ready for the development team, or to enable them to start working on different parts of the system.” — *Developer, P5*

Just as with the Refinement and Looking Ahead strategies, user feedback provided changes that were scheduled into the work for the following iterations:

[P5.445] “After the initial release, there is a lot of input from the users. This input is reacted on fast and the changes are made accordingly.” — *Developer, P5*

[P3.291] “We have parts of the user interface that we know needs improvement, and a lot of that is based on the feedback that we get, so then we will schedule it in.” — *Developer2, P3*

Implementation

The interaction design occurred at the same time as the implementation of the system. Development progress was not dependent on the design of the UI and UI design progress was not dependent on the system implementation:

[P5.412] “The HTML prototype is done in parallel to the development.” — *Developer, P5*

P3.289] “You might see how it [the UI] looks and you go back and change it and then leave it for a while.” — *Developer2, P3*

[P3.290] “Yeah, we can do several coding iterations and not worry about the UI and can come back to it . . . it can just sit there while we’re working on the code and we might go back to it.” — *Developer2, P3*

As the developer from P3 describes in the above quotes, a result of Parallelisation is that at any time during the development effort, the UI design may not correspond with the implemented system, i.e., the features included in the UI design may not be the set of features that are implemented in the system.

Evaluation

Evaluation of the interaction design was not a planned activity for P3 and P5. Acceptance testing or demonstration sessions were not explicitly seen as opportunities to test the interaction design, but happened to provide interaction design feedback.

[P5.449,450] “UI testing is basically done after development during the acceptance testing phase. We have no set [UI] test cycle.” — *Developer, P5*

[P3.260] “We do acceptance tests, basically just to say that the user interface hasn’t changed, the implementation of the user interface hasn’t changed. ” — *Developer1, P3*

UI testing for P3 occurred in a more ad hoc manner — testing a feature as it was completed by the developers, as part of the code testing, but not necessarily at every iteration:

[P3.254] “You’ll design something and pin someone down, as part of your normal work.” — *Developer1, P3*

[P3.289] “You write something, unit test it and change it, you go back and you acceptance test it, you go back, or you put it on the test simulator on the machine and see how it works and then use the user interface and you might see how it looks and you go back and change it and then leave it for a while.” — *Developer2, P3*

Due to the nature of this implementation strategy, there was the need for a synchronisation point where the UI design became integrated with the implemented system. P3 found acceptance testing to be a convenient point. P5 talked about a stabilisation sprint held towards the end of the release cycle:

[P5.453] “We reserve time towards the end of the planned release cycle for a stabilization sprint for more extensive system testing to take place.” — *Developer, P5*

[P3.254] “The task based testing we generally try and plan for that and try and fit that in at some stage and that can actually be quite different from the time you do the work.” — *Developer1, P3*

Fixes resulting from the evaluations were treated in the same way as the teams doing Refinement and Looking Ahead:

[P3.255] “Sometimes the fixes are simple and sometimes we’ll just make them. If there’s reasonably significant changes we want to make, they just get planned out as new stories, basically.” — *Developer1, P3*

P3’s developers obtained feedback by visiting their users on-site. These visits were not scheduled as part of the iterations, but occurred regularly:

[P3.262] “We don’t do it [follow up with users] as a formal process, we do it as an informal process. All our software developers tend to go on-site to visit real users.” — *Developer1, P3*

6.4.1 Key conditions

From the above discussion we infer that inherent in the development strategies that combine Comprehensive Design and Evolutionary Design with Parallelisation are two key conditions that allow teams to develop software using these strategies:

- Interaction design and agile development progress independently, and
- Changes to either the interaction design or the system implementation can be incorporated with little difficulty.

The interaction design and agile development iterations occurred in parallel. There was little or no reliance on each other in order to progress and the number and length of the interaction design and agile development iterations were also independent of each other. The interaction design was said to undergo several iterations in the same amount of time it required one implementation iteration to complete. Therefore, the iterations rarely matched up. Interaction design did not drive development as in the Refinement and Looking Ahead strategies. Instead, user stories or small chunks of formulated tasks taken out of a requirements specifications document — prioritised during the planning stage — determined the work to be done in a development iteration.

The interaction design was designed and evaluated separately from the agile development iteration. During implementation in the iteration, there were no planning or evaluation activities specifically for the interaction design. The interaction design and implementation were synchronised at some point during development, for example, before acceptance testing, where the interaction design and implementation were integrated.

6.5 Summary

Our data produced four different approaches to design and implementation of the interaction design. We refer to these four approaches as the Interaction Design Approaches, which consisted of a design strategy and an implementation strategy. There emerged two design strategies among the teams: Comprehensive Design and Evolutionary Design. For teams who performed Comprehensive Design, interaction design happened mostly up front and then the agile development process implemented the design during the development iterations. Participants who subscribed to this view, completed their UI design such that it was a representation of the entire system under development. Teams who preferred Evolutionary Design considered the ideal combination of interaction design and agile development as a process where interaction design is blended into agile development — the interaction design activities were interleaved with the agile development activities throughout the development effort. Participants who subscribed to this view produced a UI design that only implemented the features from previous iterations and the features that had been selected for a set number of iterations ahead. We note that teams who performed Comprehensive Design did not understand it to mean that they had to forego iterative UI design; they still believed in iteratively refining their design with stakeholders up-front. Similarly, teams who performed Evolutionary Design did not believe that they had to give up doing any up-front UI design whatsoever.

There emerged three implementation strategies from the data: Refinement, Looking Ahead and Parallelisation. The Refinement and Looking Ahead Strategies were a direct result of the teams' design strategy. Those teams who performed Comprehensive Design followed their design strategy with Refinement as implementation strategy, while those teams who performed Evolutionary Design combined this with the Looking Ahead strategy. Two of the teams in our study performed their interaction design

activities in parallel with the agile development activities. This was referred to as the Parallelisation strategy and did not appear to be dependent on the type of design strategy — one team performed Comprehensive Design, while the other performed Evolutionary Design. All four Interaction Design Approaches emerged as a consequence of the issues that developing software using these strategies addressed.

Chapter 7

Inside iterations

In the same way as the iterations in agile development are seen as opportunities for feedback and change regarding the code of the system [71], so we discovered that the agile development iterations were also opportunities for obtaining feedback and incorporating change in the interaction design. This chapter discusses how the teams in our study obtained feedback about their interaction design within the structure of their agile development iterations, and how that feedback was incorporated into subsequent development. Iterations were seen as driving usability testing, which in turn affected subsequent development. Teams discovered insights into design and technology limitations as they implemented the system, and iteration planning affected interaction design in terms of what features were implemented in an iteration.

Feedback and change was not only encountered in the product under development. A re-occurring theme in the interview data was the experimentation and adjustment of the development process. All teams were constantly calibrating their development process to meet the end user's goals better and to increase the overall productivity of the interaction designers and developers on the project. The following points show a coherent picture of how interaction design and agile development can work together for considerable advantage.

7.1 Development iterations drive usability testing

One of the strong themes that emerged from the data was the relationship between iterations and usability testing. In section 2.4, usability testing was described as testing how easy it is for the end users to work with the software to achieve their goals. This is a well established part of the discipline of human-computer interaction, and there are a variety of techniques that are employed. Usability testing is an organised means of obtaining usability feedback about a system and when it is missing, the interaction design suffers:

[P6.490] “It [up-front interaction design] is helpful for getting a general idea, but I’m really missing the feedback. We do get some, like I said, from our QA and sometimes from the training courses, but no, we need more feedback on, I think it’s obvious in our application because it’s lacking a grand experience, it’s lacking a grand story.” — *Developer, P6*

Teams performed usability testing with users using unimplemented forms of the interaction design, before the design was implemented:

[P3.294] “[Up-front] I prefer to do sketches and then if you can show the sketches to customers and users and do paper prototyping then that’s fabulous.” — *Developer2, P3*

[P4.352] “Yes user testing takes place up front ...you can test with lightweight, hand-drawn paper prototypes. That’s so much cheaper than building the same system and working code.” — *Interaction designer, P4*

Teams relied on inspections by the interaction designers during implementation to ensure usability goals were being met:

[P2.67] “We also have a list of about fifty usability heuristics that have to be passed and then we also have standard navigational design rules that have to be met as well.” — *Interaction designer, P2*

[P2.69] “The designers test it on a day to day basis, give feedback back to the development team to ensure if anything was missing, that we’d write a card for it and it will be captured.” — *Interaction designer, P2*

[P4.382] “What we’re currently doing at least, is that we check the user interface from time to time, so that it’s actually implemented according to the spec. That is not done in a very systematic way.” — *Interaction designer, P4*

[P8.624] “So, every time a developer finishes a user story he can submit it to a repository, he’s got to show it to the interaction designer. If there’s two interaction designers, as long as he shows it to one of them. And then they say, ‘Yeah, ok, I agree that we met the user story,’ or that, ‘We need to change some stuff.’ Obviously it’s very quick. It takes ten, fifteen minutes sometimes.” — *Project manager, P8*

While some approaches can work with lightweight non-functional prototypes, and some focus on inspection of whether principles and guidelines are followed, the ultimate determination of usability involves actual users working with actual software. The complexity of human behaviour is such that this step cannot successfully be pre-determined, automated, or simulated. The completion of iterations were seen as valuable opportunities to test the usability of the real working software at an earlier point than would have otherwise been possible:

[P4.391] “We try to do that [walkthrough] at least once before the thing is launched.” — *Interaction designer, P4*

[P7.544] “We’ve done one [test with end users], right at the beginning, I think after our first sprint, we did one usability session.” — *Developer1, P7*

Moreover, usability testing was seen as fitting in well with the agile concept of acceptance testing, or demonstration sessions, with the customer. Since customers were not always the end users, this meant that it was the business representatives feeding back usability information to the team:

[P2.72] “We’re trying writing the interaction ATs [acceptance tests] with the component and all the tests for it at the same time, so a big old vertical slice right there. So in this way we’d be hoping to make sure that whatever build the designers get back is essentially another chunk of the story. Making it easier for them to test and get a good feel of what it does and what it’s about, stuff like this.” — *Developer, P2*

[P2.70] “Through the interaction ATs, so that’s mainly how they implement the usability as such, through the interaction ATs.” — *Interaction designer, P2*

[P1.100,101] “I think the closest that we have to a usability testing is that we accept all the features manually, so we have to just stay in the acceptance meeting then we walk through the feature manually, see if it works, and there we actually don’t mind giving a feature back to engineers, saying that it has to be implemented again, or it has to be changed because of usability. Sometimes the feature actually works, it’s there but then we see it there and we’re like no. It may be the order, it might be the screen flow, the navigation or something and we’re like, no it really doesn’t work well.” — *Engineering manager, P1*

Even when the focus was not on usability testing, the customer naturally gave usability feedback during demonstration sessions:

[P8.650] “The client is really testing the UI for usability and they’ll often tell you they’re not testing the UI, they’re testing the application, but really if there’s something wrong with the UI they will tell you, ‘The colour’s not right, the label’s not ok.’ ” — *Project manager, P8*

Releasing the software to end users may also generate feedback and become a kind of usability testing, where usability feedback from users using working software, can be incorporated into the next iterations:

[P3.256] “Some stuff we’ve done we’ve gone, ‘Well, we’re so distant from these particular users, we think we understand the problem,

7.2. USABILITY TESTING RESULTS IN CHANGES IN DEVELOPMENT¹²¹

we've created the user interface, let's just release it,' and it might be difficult for them to use, but we can get back to them and ask them how they found it and get feedback that way, by releasing the software, effectively. At the minimum they get the functionality. It gives us another feedback loop that we can go to those users and say 'Does it work for you or doesn't it work for you?' " — *Developer1, P3*

[P5.445] "After the initial release, there is a lot of input from the users. This input is reacted on fast and the changes are made accordingly."
— *Developer, P5*

But iterations also bring confusion. If usability testing is seen as valuable only when the complete system can be tested, it may not be clear where in the iterative development process it can be performed. As two participants noted:

[P1.98] "I think I wouldn't know when, in our process to perform usability testing. 'Cause the UI is designed, well, up front, at least 'til the next iteration. Sometimes we don't have the UI for the full next release, just for the next iteration. Then it gets implemented, accepted and then we are done." — *Engineering manager, P1*

[P3.266] "I think putting those feedback loops in to work out whether usability is good, is very hard. It's very difficult to organise." — *Developer1, P3*

7.2 Usability testing results in changes in development

The nature of usability testing is that it frequently finds aspects of the interaction design that need changing in order to improve usability. In particular, the complex nature of human behaviour means that this is, to some extent, unavoidable, because it is simply not possible to predict how users will behave in new circumstances. So whereas software testing

can sometimes identify mistakes in programming, and whereas customers can sometimes change their requirements, usability testing can result in changes even in the absence of mistakes or changes in requirements. This is one reason an iterative process has been universally accepted for interaction design:

[P3.278] “We had the requirements and the spec and we built that and then they [customers] were like, ‘Oh but we would like this maybe.’ We try to incorporate as much of that into the changes.” — *Developer2, P3*

[P7.539] “When we demonstrate our scenarios at the end of the sprint, our media designer will be like, ‘That’s not how I think it should work.’ Or our usability expert will be like, ‘That won’t make sense.’ Even if it’s exactly what they put down on paper, things make more sense when you’re actually using working code.” — *Developer1, P7*

While the iterations in agile development are seen as opportunities for early usability testing, so are subsequent iterations seen as opportunities to change the software to accommodate the results of the testing:

[P7.595] “If you implement even a smaller subset of an overall design, customers will have issues with what you’ve built and not like things. So you may as well let them feed the usability design that gets rolled in every month.’ So I think it’s almost better that you defer that and let the iterations and feedback take care of that.’ — *Developer2, P7*

[P2.75] “We mainly card them [fixes] and negotiate between the developers, ourselves and the domain expert when we put those fixes in, which iteration those fixes will go into.” — *Interaction designer, P2*

[P2.47] “Generally if there is a UI issue, it would fall into the category of any other development issue. We want to change x, y and z, so if it does turn out to be expensive like we want a whole new chart that does all these things in 3d and does all these extra fancy bells and whistles and things like that, which is very expensive, then we have to go away and have a huddle about it and see if we can come

7.2. USABILITY TESTING RESULTS IN CHANGES IN DEVELOPMENT¹²³

up with a score, you know, come up with a price. And that will get incorporated into the development iteration in much the same way as any other changes. So we'd have a card following that request that we stick up on the board, and then we'd reshuffle all the other cards."

— *Developer, P2*

[P3.291] "We have parts of the user interface that we know needs improvement, and a lot of that is based on the feedback that we get, so then we will schedule it in ... We'll do testing of it and incorporate those tests. We've done that in iterations. We've done testing, incorporate changes and then try it again." — *Developer2, P3*

[P3.155] "Sometimes the fixes are simple and sometimes we'll just make them. If there's reasonably significant changes we want to make, they just get planned out as new stories, basically." — *Developer1, P3*

Even with the more informal approach to usability testing, where interaction designers focused on the issue of conformance to the design specification, it did appear that this process was not simple. Although the developers approached the interaction designers with queries about how the user interaction should work, this unofficial approach appeared to allow problematic software to be deployed. The participant from P4 reflected that more systematic usability testing following iterations might indeed be beneficial:

[P4.385] "But we also would need a more systematic approach to usability testing because more often than not it happens that they [developers] have completed something and they have forgotten to ask about something and then after a while we notice that we get these strange bug reports from customers and then we realise that 'Oh crap, this is something totally different than what we made.' We try to avoid that beforehand but one thing we could do, but we probably won't since our customers haven't been very interested in such things, would be to usability test the final product. " — *Interaction designer, P4*

7.3 Iterating with working software brings insights

The previous section described that when clients and users were using real working software, they suggested changes, despite the implemented interaction design agreeing with the interaction design specification. Stakeholders interacting with working software were seen to provide better feedback to the development team:

[P3.239] “With the user interface you gather as much information as you need to do some kind of real thing and then you put it through an iteration ... then you’ve got something that people can play with and look at and then you can go through another iteration or another cycle of “Is that any good? What should we be thinking about there?” So you’ve got real working software that people can reflect on a lot better.” — *Developer1, P3*

[P6.475] “So what I do now is that I very quickly mock up something that usually has a scripted set of behaviours ... it didn’t work on real data, but it was a real application, so that way our systems engineering guys who’d gather the requirements can actually pick up a real application and play with it and see, ‘Ok, well, do I like the layout? Do I like the size of it, how it looks on the screen? Do I like how it’s displaying the information? Do I like how that transition works?’ that sort of thing. And then they usually say ‘I like this,’ or ‘I don’t.’ ” — *Developer, P6*

As the system under development became an implemented product, there were further insights on the part of the developers and interaction designers as they worked with the software. One insight relates to the developers uncovering limitations in the implementation technology, which affected the implementation of the interaction design:

[P1.106] “Sometimes even during development people realise, ‘Oh this and this doesn’t work.’ ” — *Engineering Manager, P1*

7.3. ITERATING WITH WORKING SOFTWARE BRINGS INSIGHTS 125

[P7.583] “As you implement you’ll either find limitations in the technology you’re using that just don’t allow certain behaviour or certain interaction.” — *Developer2, P7*

Another insight relates to the fact that what had been specified for the interaction design during the design stage, did not always translate well to an implemented interaction design:

[P7.582] “The implementer sees a problem that no one thought of because they see it on such a fine detailed view that they bring up a design limitation of some kind.” — *Developer2, P7*

The implementation activities also revealed that interaction designers would over-specify on paper what the developers could tackle in an iteration:

[P1.127] “Doing a little bit at a time, and not thinking about the rest it’s something, I mean, [interaction designer] had a lot of problems initially but we really had to tell him a couple of times, ‘This screen that you did for us, this is for ten user stories, all we implement is two. You can’t just leave these elements in there and just not make them work,’ ” — *Engineering manager, P1*

[P7.547] “We wanted to replace current functionality with the tool, and they [interaction designers] had all these grand notions of where we’re going with it. And it’s really pretty and fun but we just couldn’t contain it in the sprint.” — *Developer1, P7*

The final insight into interaction design as a result of working with implemented software, relates to the coding activities informing the interaction design. As developers improved their code via refactoring, this refactoring in some cases also affected the interaction design, and improved the interaction design:

[P3.315] “If the code is improved then that can in certain aspects affect the user interface as well . . . If you have code that is really coupled to the functionality in the user interface it can be a pain to make changes

... If you refactor in a way that makes the system run more efficiently and the user interface run more efficiently, then that's good." — *Developer2, P3*

[P2.82] "If you still have to make code changes when a new requirement comes in then you're not finished, essentially. Sometimes a requirement will come in that means we have to change something or add on an extra widget. Or something like that that was made available by refactoring the UI code." — *Developer, P2*

7.4 Iteration planning affects interaction design

Another theme that emerged in our interviews relates to iteration planning — determining whether elements of the interaction design would be implemented in the iteration. Implementing the interaction design depended on two activities occurring during planning meetings: breaking the system up into "chunks" and negotiating between stakeholders which chunks to implement. Although the interaction designer may have made some investment in the overall interaction design as a lightweight prototype, participants understood that in order for the system to be implemented iteratively, the prototype had to be broken up into coherent chunks that could be implemented within the time of one iteration:

[P1.178] "So we'll say, we can do 14 points in an iteration, you know, we got 20 points, OK, what are we gonna do. So OK, we'd leave these out and these two need to be rethought about a little bit anyway, so we'll do most of them and we'll put these in a little pile and then we'll put these in the next time and then we'll continue. So, for the next iteration planning, we'll take those two that didn't get implemented and we'll add some more to it." — *Product manager/UI designer, P1*

[P4.338] So we're kind of using the user interface design as the requirement for the developers ... after we've made the interface design and based on that user interface design it's possible to break it

into backlog items in Scrum and give estimates of how long this is going to take, and then we make a deal with the customers, ‘This is the amount that we think this is going to take,’ and then once they say ‘Go,’ we start implementing that.” — *Interaction designer, P4*

[P3.231] “Every week we recap that and look at what we’re going to put into this week as the work for the week, so we discuss any bugs or issues that are out there and put those into the work. So once we’ve planned it out, then generally during the week it’s up to the individuals to work on that software.” — *Developer1, P3*

One participant highlighted a potential pitfall with breaking work into chunks. From an interaction design point of view, it was important that the chunks represented a coherent, complete functionality of the system:

[P4.354] “Well, this is a result of using Scrum, that I would myself, as a user interface designer, prefer that we’d have, so that the release would be more tied to the fact that the functionality is ready because the users can not do very much with the feature unless it is complete.” — *Interaction designer, P4*

Further, the negotiation between different stakeholders determined what elements of the interaction design were developed as functional software:

[P1.177] “And then after that [the UI designer doing a walk through of the cards in the iteration planning meeting], we’ll hand the cards to the engineering manager and he’ll go through and get the estimates, and he goes through the cards yet again. It’s kind of a laborious process, but, while he’s getting engineering estimates, I [the UI designer] really have to think about it because they’re, you know, giving estimates and we’re making our plan based on this, and they don’t wanna work 80 hours a week. So, more detailed questions come up during that time. So we get all the estimates and then between the engineering manager and myself, and sometimes [another developer], we figure out what’s gonna be done in an iteration.” — *Product manager/UI designer, P1*

[P7.586,587] “First we have a demo to all the stakeholders inside and outside our organisation, whoever they may be, for that particular month’s worth of work. And after that we look at all the listed items we have on the backlog we have, and roll in all the feedback we’ve got from that demonstration and feedback session. And then we plan, based on that, what the next one should contain. What the highest priority items are, what makes the most amount of sense to implement . . . It’s a very heated debate sometimes what makes it into the next month and what doesn’t. But at least all stakeholders are considered and they all, at the end, should buy into, ‘This is the best we can do.’ ” — *Developer2, P7*

Section 6.4 mentioned that when interaction design occurred independently of the implementation, as in the Parallelisation strategy, the interaction design specification played a smaller role in the planning activities than for other implementation strategies: [P5.427] “The work phase of creating the prototype is outside of the iteration planning because the work is not cut out in tasks, which reflect actual requirements.” — *Developer, P5*

The result was that the interaction design did not necessarily match the implemented features in the system code, and therefore, the planning meetings for the development iterations had little effect on the iterative progression of the interaction design.

7.5 Experimentation and adjustment of process

Both the agile development and interaction design processes have had to be adjusted in order to make them work together, to better suit the team and to smooth out the bottlenecks that arose from their combination. Participants explained that their processes were constantly being tweaked:

[P2.51] “The development process — it’s always in development, we’re always tweaking it and changing it, trying to make it better. So, pretty

much anywhere a bottleneck sits, 'cause bottlenecks shift. As soon as you deal with one bottleneck you'll find another one and they keep moving along the way." — *Developer1, P2*

One interaction designer challenged the unfavourable view of up-front design during agile development and succeeded in adjusting the team's implementation process to better suit interaction design. He attributed his ability to bring about change to his knowledge of interaction design and the authority he had, as a result of his knowledge, within the team:

[P4.398,401] "[the process inside this company] is a thing that has changed very much and I have changed it. So what I started to do, and will continue to do, is try to make this change in the process. So this is rather new and it has been quite difficult to put design up front ... I've been giving internal training here about what this means — that user interface design is not in those terms design up front, it's designing the system ... I'm not the guy you can just walk over, or my colleague. So we really know what we're doing, we really know what we're trying to accomplish." — *Interaction designer, P4*

More evidence of experimentation is clearly seen in another participant's discussion about creating acceptance tests — when to write them so that there is a better match between the build and the interaction designers' tests, and therefore, easier for the designers to do their testing:

[P2.72] "So we tried stuff like writing the interaction ATs last. That didn't work very well. So we tried writing the interaction ATs up front and we also tried writing the interaction ATs separately from whatever component it is we're trying to test. That was not working so well either, so we're trying writing the interaction ATs with the component and all the tests for it at the same time, so a big old vertical slice right there. So in this way we'd be hoping to make sure that whatever build the designers get back is essentially another chunk of the story. Making it easier for them to test and get a good feel of what it does and what it's about, stuff like this." — *Developer, P2*

Participants attributed the success of the combination of interaction design and agile development on everyone involved being open-minded and flexible.

[P2.93] “I think interaction design and agile work very well together. But I’m not sure if it’s just because we have a very open-minded set of programmers here. ” — *Interaction designer, P2*

It was clear that both designers and developers had something to learn about the other’s domain, and without open-mindedness from all sides, making the needed adjustments would not have been possible. As seen in quote P4.398,401, the developers learned that interaction design up front is not always reckless and against the values of agile development, and that interaction designers found some interaction design up front necessary. Interaction designers learned about effective communication with the developers, in terms of conveying the logic of the UI design — for which there was no concrete artifact to represent it:

[P7.538,548] “There’s a lot more interpretation in terms of how they [wireframes] behave ‘cause these are just still wireframes. We talk about, ‘What happens when you click here?’ ... The wireframes only can convey so much. They don’t show workflows and a lot of it relies on us to communicate with the people who have made those wireframes or have the expertise, and we try and do that.” — *Developer1, P7*

[P6.473] “We just had to do mock-ups, which I felt was quite unsatisfactory, because I was trying to communicate how the user interface would behave and not what it would look like.” — *Developer, P6*

Overall, the process and the team were seen as improving with time, with fewer and fewer bottlenecks impeding development:

[P1.146] “It’s been, you know, iteratively getting better and better. We started out really rough and crude ... we wanted to get it right.” — *Product manager/UI designer, P1*

The team needed time to experiment with their ideas and learn from their mistakes:

[P3.250] “I think we’re in a spot where we kind of know what we need to be doing. It’s just a matter of doing it and having more time to play with some of the ideas that we have...There’s experimentation that we really need to do and learn from our own mistakes, type of thing.” — *Developer1, P3*

[P7.563,564] “I think our process works well, or I’ll say at least better than what we’ve done in the past ... our particular rendition of how we do things, works better than what we’ve done ... We can be doing even better than what we are now. And I think it’s again something you evolve, you try it.” — *Developer2, P7*

Across all teams, there emerged two main goals, toward which all change and adjustment in the development process was geared:

- Better meeting the user’s goals
- Increasing the productivity of both the developers and designers

User goals

[P3.247] “we’re slowly getting an impression of where we should be heading with it ... we’re slowly getting to the point where we understand where we want the usability to be at, in a sort of subjective way. And that probably will be our goal ... It’s really people’s emotional reactions to the software we want now, we want them to feel that it’s a good tool and they don’t really have usability issues - they don’t complain about usability issues. I think that’s kind of where our goal is.” — *Developer1, P3*

The participants interviewed, valued the user experience of their products and adjusted their development processes in order to better meet end

users' goals. Their teams valued having access to the customer, improving the quality of their interaction with the customer, and obtaining their feedback:

[P2.60,68] "From our point of view, when users use our software, we want them to be wowed by it, you know, we want them to be astounded by how simply it meets their goals, how simply it allows them to get their work done ... In the beginning it may have been hard to meet the user's goals. In the beginning we might not have defined them well enough because it was hard to access the customer in the very beginning, but now, one thing we are doing is meeting the goals of our users." — *Interaction designer, P2*

[P8.661] "We needed more client interaction, we needed to talk in the client's language and not in UML or anything like that. So I think a lot changed." — *Project manager, P8*

[P3.262] "All our software developers tend to go on-site to visit real users. We try to keep a nice feedback loop. We can't see every user but we make sure that people experience people in the real environment, get to talk to them and ask them questions." — *Developer1, P3*

Developer and designer productivity

The teams also aimed to improve the productivity of their developers and interaction designers. Productivity gains were experienced when bottlenecks were eliminated from the process and developers and designers could proceed with their work:

[P1.172] "It's gotten a lot better lately because we've just gotten a lot better at this [acceptance testing]. Initially, user stories would go back to the engineers, like, five, six times, and that was definitely a huge bottleneck because we have this backlog of unaccepted user stories. So it was kind of a problem." — *Product manager/UI designer, P1*

[P1.174] “Typically, before the iteration there has been some up front design done. As much as possible we try to get stuff done as early as possible. So, we can do these high-level estimates and get those out of the way. We actually try to do that while we’re in the development cycle for the current release. So we’ll try and front-load it as much as we can ‘cause we don’t want to be the bottleneck and engineering wants to book releases back to back and not have a lot of lag time. So, as much as we can, we’ll try to flesh out the user stories and the user interfaces at that time.” — *Product manager/UI designer, P1*

[P4.337,338] “What we’re trying to do is actually get the user interface design out of the iteration cycle, so that we would have specified the functionality and how the user interface exactly works and the iterative part would be actually implementing that ... because the idea of trying to come up with the user interface design while doing the same piece of software has proven that it is simply impossible.” — *Interaction designer, P4*

The same participant supported their team’s change to a Comprehensive Design strategy with their experience that the negotiations with customers about the interaction design during the agile iterations held up development work:

[P4.338,347] “we make a deal with the customers, ‘This is the amount that we think this is going to take,’ and then once they say ‘Go,’ we start implementing that. Doing user interface design inside these iterations has proven that it’s simply not possible ... We try to actually push this so that this would have been done before hand, so that the features that the developers were working with, the work would have not ended and we would not have needed to put this project on hold, but our situation was such that our customers did not believe that this was going to happen.” — *Interaction designer, P4*

Developers and interaction designers experimented with tools and techniques to find the best fit between the work they were trying to do and the

cost of using those tools or techniques. The timing of when to use which technique was also mentioned:

[P2.72] “So we tried stuff like writing the interaction ATs last. That didn’t work very well. So we tried writing the interaction ATs up front and we also tried writing the interaction ATs separately from whatever component it is we’re trying to test. That was not working so well either, so we’re trying writing the interaction ATs with the component and all the tests for it at the same time, so a big old vertical slice right there. So in this way we’d be hoping to make sure that whatever build the designers get back is essentially another chunk of the story. Making it easier for them to test and get a good feel of what it does and what it’s about, stuff like this.” — *Developer, P2*

[P3.237] “Mainly we’ve played with those [UI design] techniques and over time certain ways of using them and at certain times don’t tend to work and some work better . . . part of it is working out what valuable information you get from it, so sometimes things like paper prototyping is not at all helpful because the information it’s trying to give you is not what you need — like if you’re looking for something a little more specific.” — *Developer1, P3*

7.6 Summary

In this chapter we concentrated on how teams obtained feedback about interaction design within the structure of agile development iterations. Our study identified several themes that emerged in combining interaction design and agile development:

- Development iterations drive usability testing
- Usability testing results in changes in development
- Iterating with working software brings insights

- Iteration planning affects interaction design

These points show a coherent picture of how the processes can work together for considerable advantage. Agile development iterations were found to be opportunities for generating feedback about the interaction design and for incorporating that feedback back into the interaction design. Further, iterations were seen as driving usability testing, which in turn affected subsequent development. Teams discovered insights into design and technology limitations as they implemented the system, and iteration planning affected interaction design in terms of what features were implemented in an iteration.

Experimentation and adjustment of the development process was a re-occurring theme in the interview data and was geared toward better meeting the end user's goals and increasing the overall productivity of the interaction designers and developers on the project.

Chapter 8

Agility and the interaction designer

This chapter discusses the role of the interaction designer on an agile team, as it emerged from the interview data. Two major insights obtained from participant data were that interaction design in an agile context is considered a collaborative effort between interaction designers and developers, and that the amount of collaboration depends on whether the teams consider themselves as composed of generalists or specialists. Based on these insights, we discuss the implications for the interaction designer role.

8.1 Usability has priority

All teams were convinced that developing usable products was a major priority:

[P3.244,246] “We have it [usability] in our company vision, or business plan, usability is one of those issues we need to focus on.” —
Developer1, P3

The teams showed that they had definite aims for their software in terms of the user experience:

[P2.60,70] “From our point of view, when users use our software, we want them to be wowed by it, you know, we want them to be astounded by how simply it meets their goals, how simply it allows them to get their work done. And we also have the belief that good design is invisible. When you’re using the software you shouldn’t actually notice design. It should just be “My god I can sell my product this way, my god it’s allowing me to open the report and customise it the way I want.”... the usability will be built into the design of the screens.” — *Interaction designer, P2*

[P3.247,299] “We want them to feel that it’s a good tool and they don’t really have usability issues — they don’t complain about usability issues. I think that’s kind of where our goal is ... we don’t have a formal usability plan, but we have values that we are striving and working towards.” — *Developer1, P3*

[P3.302] “I wouldn’t say that they’re [usability goals] achieved, because that word indicates that they’re perfect in our system, but we’re striving for them.” — *Developer2, P3*

[P4.376] “[With interaction design] there are two things. One is utility, which is you can do it with this piece of software, and that is the primary thing that we try to accomplish. The second thing we try to accomplish is usability and in usability, because we build tools for people, we pick first efficiency and learnability comes only after that. That is the way we design things well.” — *Interaction designer, P4*

The benefits from usable products were not only experienced by the end users, but also by the developer company. For the company, usable products had a positive impact on their Return on Investment and repeat business. For the participants, usability was seen to involve caring about solving users’ goals and creating a good user experience. Good usability was considered to depend on how well the team understood the goals of the customer and how well their product met those goals:

[P2.91] “Putting the front end together is really just the final part of the process. It’s going in and talking to the customers, understanding

what their needs are, understanding what will fit into their current sales process. There's no point in going ahead and building a tool that actually won't fit into any of their current processes. You have to really understand what they're doing, how they work, what their future direction is so that what we're bringing into them fits very comfortably into their system that they're using, or into their sales process, or into their environment, or into their future directions, or otherwise it'll just be a tool. And we won't get that user satisfaction that we're looking for or repeat business. So it's more than just about screen design, it's just the whole process of understanding what a customer needs and making sure that we meet all the different goals that they have." — *Interaction designer, P2*

The only instance where usability of the product was seen to be secondary was in the case of P3, who considered the functionality their product provides to their users as the most important, highly valuable (compared to usability of those functions) and in some instances superior to the competition's. Therefore, it was their opinion that it is only necessary to spend that amount of resources on usability that ensures competitive advantage — even if that means that some usability issues still exist in the product. The team did intend to fix the usability issues in subsequent releases:

[P3.245] "With our system the functional aspect of the software is hugely valuable and in some cases our software does what other people's software can't and if it was the hardest thing to use, they would employ the smartest person they could find to use it, and they would save themselves millions of dollars." — *Developer1, P3*

[P3.246] "The big thing with these kinds of products is you're really looking to be competitive, so, all you have to be is more competitive than your competition, so, there's a certain amount of usability that we want in our software and we want to make it reasonably easy but there's no point in investing in making it perhaps as usable as some of the consumer products out there because we get no more competitive advantage. We want to invest to make sure that we're getting a good

return for our money and leverage as much stuff as possible, but we probably wouldn't go to the extent of some of the laboratory testing some people are getting into these days, tracking eye movements to get the finer details. And we could probably make our product a lot more usable if we went to that degree, but we're looking for sort of a low lying fruit. Because you know there are a lot of mistakes in our usability and fixing those up and having feedback and processes to test that, we could get a lot of benefit just from that." — *Developer1, P3*

8.2 Interaction design requires skill

The second insight into interaction design in an agile development context, was interaction design being considered by the team as more than adding a front end to a system — more than a mere add-on to a product. In particular, interaction design was seen to improve understanding of the users, their goals, and the context in which the product was to be released. Teams who followed a Comprehensive Design strategy, valued the opportunity to design the functionality of the whole system in a fast and cheap way:

[P2.91] "Putting the front end together is really just the final part of the process. It's going in and talking to the customers, understanding what their needs are, understanding what will fit into their current sales process. There's no point in going ahead and building a tool that actually won't fit into any of their current processes. You have to really understand what they're doing, how they work, what their future direction is so that what we're bringing into them fits very comfortably into their system that they're using, or into their sales process, or into their environment, or into their future directions, or otherwise it'll just be a tool. And we won't get that user satisfaction that we're looking for or repeat business. So it's more than just about screen design, it's just the whole process of understanding what a

customer needs and making sure that we meet all the different goals that they have.” — *Interaction designer, P2*

[P2.90] “It’s essential. For anything that isn’t going to be exclusively used by programmers, you’re going to need someone who knows how to put a UI together. And not just because it will be pretty and easy to use and stuff like that but because the kind of bottlenecks and kind of delays and waste you get without having someone who knows how to put a front end together will add to the cost at the end of the day.” — *Developer, P2*

[P4.335,396] “It [interaction design] is more about designing functionality than only the user interface, but it’s very easy to design the whole functionality of the system, since it has no kind of motor underneath that would not be linked to the user’s tasks. Everything serves the users and the system, so the functionality is the most easy to design through user interface design . . . It almost insults me to say that we’re trying to do something hi-fi. The fact that user interface design, or the user interface, is something that you can just add on to the system. Because what we do is we actually design the functionality of the system and that’s definitely not an add on, it’s definitely the system we’re designing — from the user’s point of view, not of course the things under cover.” — *Interaction designer, P4*

Interaction designers were seen to possess an intrinsic knowledge about usable design, interaction design techniques, as well as using and interpreting feedback from usability testing:

[P2.27] “It’s just basically good common sense, good design rules and as I said, a lot of influence from the Cooper book *About Face 2.0*.” — *Interaction designer, P2*

[P6.488] “The benefit of having somebody who’s been trained in the patterns, who can recognise it and go, ‘Look, you know what, that’s just going to cause this reaction.’ So, like, in programming, we have design patterns that are based on certain architecture, so obviously

user experience is just as easily related to architecture, so, even if they're not codified there's probably a pattern that somebody like that can recognise." — *Developer, P6*

[P2.67] "We have a lot of goals and heuristics that have to be met and at this stage it's ingrained, you just know, you don't have to go and physically check through a list. You just know, by using it, by looking at a screen you know whether it's right or wrong." — *Interaction designer, P2*

[P1.190] "I started out trying to design backwards from Jakob Nielsen's usability heuristics. People don't really do that a lot, but I just try to keep those things in mind. I had some usability activities that I wanted to accomplish, I had some goals." — *Product manager/User interface designer, P1*

[P1.130] "We have things like experience from previous projects, we have things like design factors." — *Engineering manager, P1*

Individuals in the interaction designer role were also seen to possess the necessary skills to elicit requirements and feedback from the users, during user research activities, and for modeling the knowledge about those users, in the form of personas and scenarios:

[P2.66,18] "We design personas up front for our projects and we identify what their goals are, in using the product. We have to make sure that when we're testing the product, we meet all their goals ... In interaction design we write our user stories through what are called scenarios. Before they ever become user stories on a card for the development team, we do them via scenarios." — *Interaction designer, P2*

[P1.122,151] "We created some personas. And so our personas represent different users of our system ... And for the UI designer, I mean, he really had to think, ok, now I have these, let's say fifteen user stories for our task manager; seven of them are for the developer, three are for the project manager, five are for the business analyst. So then

it would be up to the product manager to make sure, ok, which of these individuals has the full feature set — there's nothing missing. And then for the UI designer, he just took these stories and then, again, really, with the persona in mind, try to implement it as quick as possible." — *Engineering manager, P1*

[P1.118] "I mean that was for the UI, but also for the user stories, one of the most important things to do. To really have these personas ... It helped us a lot for the UI just for the simple fact that every now and then we're talking about a certain user story how to implement it. It just helped us to go back to the user story and see, ok, for which personas or which user is this — is this for a developer, is this for a manager, is this for a QA person, for a business analyst. The implementation, and even the UI implementation, changes. Thinking, oh, ok, for a developer you can have a very technical screen with lots of data — developers don't mind. But if it's someone as simple-minded as me, as a director or even higher, people on that level want to have a much simpler UI they don't want to have lots of detail." — *Engineering manager, P1*

[P2.16,17,20] "We'll hold a workshop with the project sponsors, we'd have them with end-users of the product, we'd have them with IT people, with the actuaries in the company, with the compliance people, as many as we can who will have input into the product or are using it in some form, or who are developing it in some form. They come from our workshop. We gather the user stories via that ... Sometimes we would try and sit with an end user and do maybe cognitive walkthroughs, where we get the user to talk through what they're doing in their current day to day with their current piece of technology. We try to understand how they currently use technology and their frustrations with it." — *Interaction designer, P2*

In one case, the interaction designer was the customer proxy:

[P1.199] "I end up being, like, a proxy for the customer. So I'm taking into account customer requests and things like that, and some

feedback.” — *Product manager/User interface designer, P1*

8.2.1 When developers do interaction design

Participants felt that interaction design work should not be done by developers, due to their training and background being in a different domain:

[P1.201] “And I don’t think the user interface designers should be engineers, ‘cause I think that’s an inherent problem because if you’re focused on writing code and not building user interfaces, that would change things.” — *Product manager/User interface designer, P1*

[P2.56] “I can’t imagine why a developer would be designing screens because their training lies in a whole different area to me, and their skill set lies in a different area so I just don’t understand how companies can do that.” — *Interaction designer, P2*

[P2.64] “Programmers do not make good designers, they just don’t. Anytime I design anything at all it always comes out in horrible blues and greens and there’s not an awful lot I can do about that, and programmers being very logical people, you know, will always say that if they haven’t any experience of design then it just doesn’t exist. You know, it is very difficult to get people out of their own little world, that way.” — *Developer, P2*

[P2.97] “The UI doesn’t naturally fall inside my domain, it isn’t something I naturally do well. I can make UIs at the drop of a hat but they won’t look good, they won’t be in any way what the customer wanted, so why am I doing it?” — *Developer, P2*

[P6.509] “Maybe more people need user experience designers – even if they were programmers – because that’s somebody focusing their brain in a different direction, and it may just be that somebody can’t wear both of those hats.” — *Developer, P6*

Developers who were not trained in interaction design techniques at times thought that usability was not their responsibility:

[P7.602] “They’re not held accountable for the usability portion of it ... as their traditional role dictates, they’re implementers. They’re not business experts, they’re not usability experts. They don’t necessarily have fine details as to what their end consumer’s going to do day in day out with this product.” — *Developer2, P7*

[P1.188] “The guy who’s coding, you know, he’s not a user interface expert. He doesn’t know a lot about usability testing. Probably doesn’t think it’s his responsibility.” — *Product manager/User interface designer, P1*

Participants felt that developers should not be designers, as developers tended to miss the subtle interaction design issues that interaction designers are trained to recognise:

[P1.173,201] “The implementation of the design was difficult and sometimes people inadvertently cut corners, like, they’d look at it and to their eyes it would be good enough and then, from a designer’s perspective, it’s like no that’s not, you know, that radio button isn’t selected by default, or, like that text string is misspelt, or you’re using the wrong term there ... They probably didn’t even see it. So it’s just complexity of the UI itself ... I don’t think the user interface designers should be engineers.” — *Product manager/User interface designer*

[P3.300] “There still can be team members who look at a screen and say, ‘Well, this is really easy and straight forward to use,’ and then you’re [an interaction designer] like ‘No.’ ” — *Developer2, P2*

8.2.2 Interaction design skill contributes value

Participants valued interaction designers for bringing a different type of skill and knowledge to the team — skills that the developers did not have. Therefore, teams unanimously agreed that interaction designers were a valued role and that they should be part of the team:

[P2.28] “We have four developers and two UI specialists, so I definitely do support the idea of having them [interaction designers] in a

team ... I can't imagine us not having interaction designers involved in the design of the product." — *Interaction designer, P2*

[P8.656] "The [user] satisfaction is very high, customers' expectations are being met, and I'd say the main reason is because on that we team we probably have the two best interaction designers." — *Project Manager, P8*

8.3 Interaction design as collaboration

Throughout the interview data, there emerged an awareness within the teams that agile development influenced the interaction designer role in ways that were considered different to the traditional view of the role. The frequent and open communication between interaction designers and developers during the development effort, resulted in an activity which was seen as more collaborative and receptive to developer feedback than traditional interaction design:

[P1.106] "But it's a fantastic thing; you have these sometimes very opinionated discussions up front and sometimes even during development. I mean, people realise, 'Oh this and this doesn't work,' and then they go to the UI designer. You know, it's just two different opinions: you can do this or this. I think it's fantastic." — *Engineering manager, P1*

[P1.111] "The other thing, I think, you don't need that many UI designers in XP because the relationship between the UI designer and the developer is just very different. It's not that the UI designer has to come up with the full UI design, everything final, all JavaScripts and everything in it. Then goes to the engineers and says, 'Ok, implement this,' and then sees it four months later and has to, like, look through it how does it work? With us it is much more, like, day to day communication. The UI designer actually can get away with not putting all the details and everything into it. Many things just work out dur-

ing the iteration planning or during development.” — *Engineering manager, P1*

[P1.112] “In the future comes the developer and he says, ‘Ok, maybe we actually do it like this, maybe we use that colour, or maybe we just do this element or that element.’ And that was something that [interaction designer] really had to get used to. He initially found it as a loss of control. He doesn’t control everything in the UI.” — *Engineering manager, P1*

[P3.297] “We see each other every day and we sit all together and even working on different projects and different stories, we talk to each other all the time, ask each other’s opinions for the user interface and the code.” — *Developer, P3*

8.3.1 Setting the target

Interaction designers across all projects were setting the standard for the usability of the product for the rest of the team, whether in the form of usability goals, values or UI design specifications:

[P1.187] “So, what I would try to do is get a high level understanding of what the product’s gonna be, and I’d probably try to work some things out. Just create some up front consistency, like, what do buttons look like, where are they placed, what do tables look like, how do users interact with tables, what do forms look like, how do you get from a table to a form and then back to the table, like, basic interaction models. So, kinda like a style guide.” — *Product manager/User interface designer, P1*

[P4.358] “For design we start with pen and paper. We do paper prototypes ... What we try to accomplish currently is that we get this HTML directly to the implementers.” — *Interaction designer, P4*

The UI design, whether in the form of a prototype or a specifications document, was an important tool for creating work and cost estimates used in the planning of the implementation of the subsequent releases:

[P4.356] “From my point of view, the user interface design quite heavily leads the release planning” — *Interaction designer, P4*

[P5.427,428] “So the work of the prototype has to be planned so that we can discuss what parts of the system should be developed first, so the development team can start working on them. So we try to come up with a list of actual features that are broken down into individual tasks that then can be estimated and it follows a basic Scrum process, from then on, that developers give estimates and then at least the customer’s project manager, and preferably also the business people are present when doing prioritisation within the different features. And then based on the estimates given on the different tasks involved in each requirement, a list of tasks or backlog for that sprint is forged out and then everybody gets to pick their own tasks and work is started.” — *Developer, P5*

After the tasks had been formulated and written on to story cards, the interaction designer was involved in negotiations about which cards were to be implemented in an iteration:

[P2.43] “We would negotiate between the developers and the domain expert if whether any new cards would go into the iteration.” — *Interaction designer, P2*

[P1.177] “And then after that [the interaction designer doing a walk through of the cards], in the iteration planning meeting, we’ll hand the cards to the engineering manager and he’ll go through and get the estimates, and he goes through the cards yet again. It’s kind of a laborious process, but, while he’s getting engineering estimates, I really have to think about it because they’re, you know, giving estimates and we’re making our plan based on this, and they don’t wanna work 80 hours a week. So, more detailed questions come up during that time. So we get all the estimates and then between the engineering manager and myself, and sometimes the engineering manager, we figure out what’s gonna be done in an iteration.” — *Product manager/User interface designer, P1*

8.3.2 Maintaining the target

During the development effort, it was usually the responsibility of the interaction designer to continually verify that what the developers were implementing was acceptable in terms of usability. During the development iterations, the interaction designer was required to be on hand to answer developer queries about the design and identify usability issues missed by the developers:

[P4.343] “There is a need for a UI designer even still while the software is being implemented because you get these kinds of feedback and problems and technical limitations and redesigns needed and the user interface designer should be, if possible, part of the team.” — *Interaction designer, P4*

[P4.382,384] “What we’re currently doing at least, is that we check the user interface from time to time, so that it’s actually implemented according to the spec. That is not done in a very systematic way. They [developers] say, ‘Well, look, come and check this out,’ so we go and check it during the implementation.” — *Interaction designer, P4*

[P7.577] “I would say the user experience person is officially the one who’s responsible for saying, ‘Yes or no,’ a good interaction versus a bad interaction.” — *Developer2, P7*

[P4.384] “We also do this unofficially that they [developers] say, ‘Well, look, come and check this out,’ so we go and check it [UI design] during the implementation.” — *Interaction designer, P4*

[P1.106] “During development. I mean, people realise, oh this and this doesn’t work and then they go to [the interaction designer].” — *Engineering manager, P1*

When the team was constrained by resources such as time, the interaction designer became the main driver for usability — by trying to incorporate as much usability techniques into the allowed time frame:

[P3.273,298] “If sales go out and sell a feature and then we have to deliver it within a time frame and then that’s the time you’ve got. But I try to do as much as I can and I try to incorporate as much usability as I can ... I think the benefits of usability are clearly seen and, of course we should do more and it would be great if we had more time ... you only have that much time and that much resources. You try to do the best with what you’ve got.” — *Developer2, P3*

During implementation, the interaction designer may have opportunities to test the user interface during acceptance testing. The interaction designer may also accompany the rest of the team in a manual walk-through of the features, concentrating on identifying usability issues. If any testing with users had to be performed during the implementation process, the interaction designer was responsible for running those testing sessions:

[P2.43,58] “We have acceptance tests written by the designers, we have acceptance tests written by the domain expert ... We specify how the interaction navigation works via acceptance tests. So that’s really how we specify the navigation, the rules, the error checking and all that.” — *Interaction designer, P2*

[P1.100] “We accept all the features manually, so we have to just stay in the acceptance meeting then we walk through the feature manually, see if it works.” — *Engineering manager, P1*

[P4.390] “We try and do the evaluations — it’s kind of a walkthrough. We have paper prototypes, written-in data and then we have a goal or task and then we simulate how the user interface works for the user. Sometimes we ask them ‘What would you press here?’ but that’s not actually required in those cases. We’re not interested in those kinds of learnability issues, we’re mostly interested in missing functionality or if we understood any concepts wrong. We call this a utility walkthrough. It’s not a term from the literature, we just developed it on our own. ... We try to do that [utility walkthrough] at least once before the thing is launched, but sometimes there is no budget

8.4. LURKING PITFALLS OF THE INTERACTION DESIGNER ROLE¹⁵¹

for that, so we just have to go on with what we have.” — *Interaction designer, P4*

Along with the developers, the interaction designers decided on whether a build was complete, or whether more design or redesign was necessary:

[P2.42] “And the designers take builds every morning. Because we have a very tight development cycle, we have builds hourly. So we take the builds every morning, check the cards that were done yesterday and the designers can sign off on those cards to make sure they’re happy with the changes that were made. And so we can kind of go back and forth between each other until we’re both happy.” — *Interaction designer, P2*

[P4.383] “What it should do is we should get some kind of a ticket, I mean, this ticket in our problem handling system, kind of the bug database, so that it’s kind of moved that I’ve now done this feature and that it’s assigned to the user interface designer, so that he or she should check it through before it’s said that this is now ready.” — *Interaction designer, P4*

[P1.101] “There [during acceptance testing] we walk through and there we actually don’t mind giving a feature back to engineers, saying that it has to be implemented again, or it has to be changed because of usability. Sometimes the feature actually works, it’s there but then we see it there and we’re like no. It may be the order, it might be the screen flow, the navigation or something and we’re like, no it really doesn’t work well. ” — *Engineering manager, P1*

8.4 Lurking pitfalls of the interaction designer role

We identified two situations that could potentially pose challenges and be somewhat problematic for the interaction designer role. The first concerns

the case where interaction designers are also developers and the second pertains to the authority of the interaction designer on the team. In both cases the designs produced by the interaction designer may be compromised unless attention is specifically given to these situations:

When UI designers are developers it was found to be important to consciously try not to think about implementation details when designing the UI — that resources and technical issues do not constrain interaction design:

[P3.286] “Maybe a disadvantage of being a programmer and an interaction designer is that you get too close to the programmatic side of thinking and maybe too understanding of the constraints in resources and technical issues.” — *Developer2, P3*

[P3.229] “We would go for the simplest to implement UI rather than perhaps the most usable UI.” — *Developer1, P3*

The team must also take care that interaction design issues do not get prioritised away in favour of coding issues:

[P3.281,282] “I think it’s [the UI and its code] owned by the team but valued differently. It’s everybody’s responsibility, but with the different backgrounds and focuses and interests, people will value it more or less ... I think it’s necessary to have that specialised [interaction] knowledge in the team because, I think, if you don’t have it it’s easy to get the attitude that it’s just basically common sense. They get prioritised away.” *Developer2, P3*

The level of authority of the interaction designer within the team could determine the success with which the interaction designs are accepted by the other team members. One participant related a story contrasting his experience with that of another interaction designer who did not have his level of expertise in interaction design:

P4.401] “I’ve been developing design methods with my wife at the university, evaluation methods and such, I’m not they guy you can just walk over, or my colleague. So we really know what we’re doing, we really know what we’re trying to accomplish, so you can’t just push us around, which was the case we had in this company one girl that did user interface design with no training but someone had to do it. She was very easy to push around and redesign the user interfaces after she’d done them because she had no authority. She knew that this is not her field of expertise, she could not hang on very tightly to what she had done. She knew that this might not be the best thing. Whereas we know that what we do is absolutely great so there’s no walking over us!” — *Interaction designer, P4*

8.5 Interaction designer: Shared vs. total control

At this point the generalist/specialist aspect of agile teams should be noted. The literature suggests that agile teams are ideally generalists, i.e., they possess a wide range of skills rather than one narrowly defined area of expertise [70, 93]. If this view extends to the interaction designer and developer roles as well, then agile team members with interaction design skills along with technical development skills would be preferred over exclusively specialised interaction design skills or exclusively specialised development skills. In our study we encountered evidence of both agreement and disagreement with this view. Table 8.1 shows which teams distinguished between the roles of developer and interaction designer.

P3 and P6, both XP teams, were the only teams who explicitly stated that there were no distinctions between the roles of interaction designers and developers in their teams, i.e., they were all considered developers. However, as will be evident in the rest of the discussion, P1 strongly tended toward a generalist approach and so when reference to generalist teams

Team	Agile	Interaction Design Approach	Roles
P1	XP	Evolutionary Design with Looking Ahead	yes
P2	XP	Comprehensive Design with Refinement	yes
P3	XP	Evolutionary Design with Parallelisation	no
P4	Scrum	Comprehensive Design with Refinement	yes
P5	Scrum	Comprehensive Design with Parallelisation	yes
P6	XP	Comprehensive Design with Refinement	no
P7	Scrum	Evolutionary Design with Looking Ahead	yes
P8	XP	Evolutionary Design with Looking Ahead	yes
P9	XP	Comprehensive Design with Refinement	yes

Table 8.1: Team and Roles

are made in later discussion, this will include P1. A participant from P3 explained the preference for generalists:

[P3.222] “I tend to avoid the word specialist because I don’t want somebody who’s a specialist. I want an expert on the team rather than a specialist. So I expect them to understand the code and what not.” — *Developer1, P3*

The same participant went on to state that the whole team had been sent on an interaction design course to raise their awareness of usability issues:

[P3.224] “And we’ve also done some training for everyone on interaction design ... that’s the kind of knowledge we want everyone on our team to slowly pick up their ability in doing and being aware of what issues are involved.” — *Developer1, P3*

An interesting contradiction occurred for P3 where the one participant, quoted above, preferred a team of generalists, the other participant from P3 highlighted the need for specialisation:

[P3.282] “Yes I support the notion of having interaction designers as part of the team. I think it’s necessary to have that specialised knowledge in the team because, I think, if you don’t have it it’s easy to get the attitude that it’s just basically common sense.” — *Developer2, P3*

For the rest of the teams who distinguished between interaction designer and developer roles, the preference for specialists was clear. The exception was the engineering manager from P1, who expressed the view that interaction designers should tend towards generality, i.e., that the interaction designer should ideally possess a certain level of technical knowledge, as well as skill in interaction design, in order to ensure an optimal front end design that matched the technology of the back end:

[P1.109] “We want to hire a new UI designer. We want to get someone who is much more technical ... who is still not a full time engineer, but who actually can give [interaction designer] a lot of feedback, a lot of ideas. Say if you wanna do this, look, you could use this and JavaScript or we could do this or we could do that ... where he just knows, hey, these are all the technical things that I could do, how can I make my UI design better.” — *Engineering manager, P1*

The generalist and specialist teams differed most notably with respect to the amount of control the interaction designer was seen to have over the interaction design of the product. The teams of generalists felt that the developers were also able to come up with valid interaction design and advocated shared ownership of the UI design, in the same way that they advocated shared code ownership.

[P3.219,220] “In terms of the code it’s definitely shared. It’s common code ownership XP-style. The user interface, that’s effectively common ownership as well.” — *Developer1, P3*

[P1.176] “We’ll take the user interface and the user stories into an iteration ... and people will start asking more detailed questions, challenging some of the ideas.” — *Product manager/User interface designer, P1*

[P1.112] “In the future comes the developer and he says, ‘Ok, maybe we actually do it like this, maybe we use that colour, or maybe we just do this element or that element.’ And that was something that [interaction designer] really had to get used to. He initially found

it as a loss of control. He doesn't control everything in the UI." — *Engineering manager, P1*

[P1.188] "I think usability is the responsibility of everybody, not just the developers and if your XP process is insulated to just the engineering team then I think that's probably the wrong approach. If you don't have product management on board, if you don't have all stakeholders involved with XP, then you're in a bad situation, 'cause the guy who's coding, you know, he's not a user interface expert. He doesn't know a lot about usability testing. Probably doesn't think it's his responsibility" — *Product manager/User interface designer, P1*

[P1.158] "You typically go through the functionality with the engineers and the engineers will poke as many holes in it as they can to really get their heads around it and, you know, a lot of times I'll end up making tweaks or sometimes mass changes to the UI based on an assumption I made that really doesn't work." — *Product manager/UI designer, P1*

[P6.469] "Any programmer who works on the user interface portion of our code will largely be responsible for that and they are responsible for the design as well." — *Developer, P6*

The teams of specialists felt that interaction designers should have full say in interaction design matters. P2, P4, P5, P7, P8 and P9 preferred to delineate the responsibility of the interaction designers as being only the interaction design, since having interaction designers and developers concentrate on their own domain makes their work more efficient:

[P4.341] "But actually the current status, and I think that it's very good, is the fact that the user interface designers do the design, it's their responsibility, because in this project, this has gone well." — *Interaction designer, P4*

[P5.426] "I would say that because we've driven the project so that the customer develops the idea and the user interface designs by themselves, and not leaking any information out of it before they get

it ready ... It makes work efficient but it can also become a bottleneck." — *Developer, P5*

[P2.32,97] "Pretty much design-wise whatever the interaction designers say goes ... it takes a lot of responsibility away from my job. If I don't have to worry about ui concerns, I can get more work done and get it done better ... From that point of view I am able to draw a line between what is my problem and what's [interaction designer's] problem, or rather what's my responsibility and what's [interaction designer's] responsibility to put it in a nicer way and to concentrate on what I do really well." — *Developer, P2*

Their UI design created before implementation began, combined with the almost total control of the interaction designers over the interaction design, ensured that the adjustments based on developer feedback was minor:

[P2.23] "There wouldn't be that many changes once it goes into the development iterations." — *Interaction designer, P2*

[P8.639] "It [UI design] does change because of technical challenges, and so on. It happens often but on minor things, right. There's a lot of tweaking on minor things, 'I can't put this here because of this or this.' " — *Project manager, P8*

Further, participants on specialist teams distinguished between the UI *design* and the *code* that implements that design, seeing the UI design as the interaction designer's responsibility and the code that implements it as shared among the developers:

[P4.342] "The UI is owned by the user interface designer, but the code underneath is owned by everybody. ... But the responsibility is clearly the user interface designer's and the code ownership is shared between those who write the code." — *Interaction designer, P4*

The teams based whether it was appropriate for developers to give interaction designers feedback about interaction design on their attitude about

which role had responsibility for the interaction design. The teams of specialists felt that developers should not give feedback on UI design issues, as they are not qualified to do this:

[P2.31] “I have to say in our team here, each of us have great respect for each other’s work. We as designers have great respect for what the developers do and the developers have great respect for us. Even with great communication going, we’d never assume to make a suggestion about something we don’t know very much about.” — *Interaction designer, P2*

[P4.397] “From the user’s point of view what the user interface is, is it’s this design and the other design not the same thing, so that if there’s a good design for that, there’s a reason why the design is good and there is no reason for the implementer to create something else, if what’s a good design is buildable.” — *Interaction designer, P4*

In the teams of generalists, discussion and debate about UI design issues were encouraged. These teams saw this as the way in which they created the best UI designs:

[P1.105,106] “Makes actually [interaction designer’s] life hell sometimes because he’s the UI designer, the product manager. He comes to the meeting and says ‘Ok, here’s the UI and here’s how we do this and this feature.’ And there’s ten engineers sitting there and saying ‘Look, nobody works like this. What are you doing? It’s like, nobody’s doing this’ and so sometimes it’s really, really hard for him to, like, tell them, ‘Look, you might not work like this but I think outside, people who use our application, they will work like this, they will appreciate this feature.’ But its a fantastic thing; you have these sometimes very opinionated discussions up front and sometimes even during development ... it’s just two different opinions: you can do this or this. I think it’s fantastic.” — *Engineering manager, P1*

[P1.159] “So, you typically go through the functionality with the engineers and the engineers will poke as many holes in it as they can to

really get their heads around it and, you know, a lot of times I'll end up making tweaks or sometimes mass changes to the UI based on an assumption I made that really doesn't work. You know, like, something I thought that would work that isn't very logical." — *Product manager/User interface designer, P1*

[P1.168] "And you know, we've reviewed it with the engineers and basically blew it to pieces and they have better ideas about this so, we need to go back to the drawing board and figure this out." — *Product manager/User interface designer, P1*

8.6 Summary

This chapter discussed the valued role of the interaction designer on an agile team. Developing usable products was of high priority for our participants, which requires a special skill that can not be easily deferred to the developers. Within the agile context, interaction design was seen to be a more collaborative activity than traditional interaction design. Interaction designers set the target for the usability of the product and were continually involved during the course of development to ensure that those targets were reached.

Pitfalls of the interaction designer role in agile development were also identified. The first concerned the level of authority of the interaction designer within the team. The second concerned the case where developers are also expected to be interaction designers. Both situations have the potential to compromise the overall interaction design.

Finally, the discussion on whether the view of generalist or specialist teams extended to the interaction designer/developer roles saw teams advocating both. On the generalist teams, there was the view that developers have valuable contributions to make to the interaction design and discussions and debates were encouraged. The specialist teams opposed the generalist views and considered the role of developer, with respect to

the UI, as simply the role of implementer — any feedback from the developer about the interaction design should only concern implementation issues.

Chapter 9

Conclusion

The intention of our grounded theory study was to investigate how real-world agile teams combine interaction design with their agile development activities. During the analysis of the data, the empirical categories that emerged, led to the findings discussed in chapters 5 to 8. In section 3.6, we identified risks that could compromise the reliability and validity of our study, and we adopted specific measures to address each risk. Even so, this grounded theory study can not be considered to be a complete sampling of the topic, and can not claim to be a complete reflection on practice in the field. As explained in section 3.1, the strength of a qualitative grounded theory study is to help develop understanding of the phenomena of the research topic by providing rich descriptions, rather than a quantitative summary.

Here we present our contributions, present topics for future discussion and suggest further work.

9.1 Contributions

Overall, the results contribute to a better understanding of the place of interaction design in agile development. Next, we present our specific contributions:

1. **Up-front interaction design happens**

For a development project to even begin, all teams underlined the importance of studying their end users and researching the scope of the product under development — especially for the teams whose development projects were organised such that their time with end users once implementation had begun was minimal, or non-existent. The teams gained a holistic view of what their product was to become, which together with the knowledge about their users, guided the rest of the development effort. Up-front design was credited with cost and time savings, increasing user satisfaction, increasing usability testing productivity, mitigating the risks of the developers' work and anchoring the UI design. The amount of interaction design completed at this time depended on what the team's approach to interaction design throughout the development effort was going to be. Certain teams completed all their interaction design up front whereas others completed just enough to get started and then evolved the interaction design during the implementation stage.

2. **Agile development can be structured by four approaches to interaction design**

Our analysis showed that the teams organised and structured their interaction design and agile development activities according to their 'Interaction Design Approach', which consisted of a design strategy and an implementation strategy for the interaction design. There emerged two design strategies — Comprehensive Design and Evolutionary Design — and three implementation strategies — Refinement, Looking Ahead, Parallelisation — that the teams combined in four different ways. The Comprehensive Design strategy, combined with Refinement, was characterised by teams completing a large proportion of their interaction design up front, designing for all the features that would be included in the final product, and following this with implementing and refining the interaction design

during the agile development iterations. The Evolutionary Design strategy, combined with Looking Ahead, was characterised by teams completing just enough interaction design to start implementation and then evolving the interaction design, along with the rest of the system, during implementation. The interaction design always included only features for the current iteration and those that had been selected for a set number of iterations ahead of the current iteration. Parallelisation was combined with both Comprehensive Design and Evolutionary Design strategies, and was characterised by the interaction design and agile development occurring in parallel. This was possible since the interaction design and the system code were not dependent on each other to progress.

3. Agile development iterations are opportunities for feedback and change in the interaction design

Within the structure of agile development iterations, our study identified four aspects that provided opportunities for feedback and change in the interaction design:

- (a) Development iterations drive usability testing
- (b) Usability testing results in changes in development
- (c) Iterating with working software brings insights
- (d) Iteration planning affects interaction design

These points show a coherent picture of how interaction design and agile development can work together for considerable advantage. Agile development iterations were found to be opportunities for generating feedback about the interaction design and for incorporating that feedback back into the interaction design. An important insight was that the combination of interaction design and agile development was the result of constant experimentation and adjustment of

process, in order to improve the productivity of the interaction designers and developers, and to meet the goals of the end user better.

4. The interaction designer role is a highly collaborative role on agile teams

Within the agile context, interaction design was seen to be a more collaborative activity than traditional interaction design. Interaction designers set the target for the usability of the product and were continually involved during the course of development to ensure that those targets were reached. The effectiveness of the interaction designer role was found to be dependent on the level of authority of the interaction designer within the team and whether developers were also expected to be interaction designers. Finally, the teams that considered themselves teams of generalists, encouraged discussions and debates regarding the interaction design. The teams that considered themselves teams of specialists, considered it appropriate that the developers only feed back implementation issues about the interaction design.

9.2 Returning to the literature

Referring back to the related studies mentioned in section 2.5.3, we now examine their process descriptions in terms of the interaction design approaches we have identified in our data.

The first relevant study is the collection of refereed papers and tutorials by Patton, in which Usage-Centered Design is combined with a typical agile development methodology [110]. Information on the amount of interaction design suggested to be performed up front is not clear, however from the description of how Usage-Centered Design can be ‘inserted’ into the agile iteration, it can be inferred that either combination of Comprehensive Design with Refinement or Evolutionary Design with Looking

Ahead would be possible.

The study by Beyer Holtzblatt and Baker highlights the need to determine the UI up front and to “separate design from engineering” [13]. This is reminiscent of the Comprehensive Design and Refinement approach described in this thesis, in which participants using this approach see the interaction design and agile development processes as separate.

In the studies by both Miller [96] and Sy [137] the interaction designers were doing detailed design for the iteration that the developers would do next, and doing evaluation of the iteration that the developers did last. This agrees with the Evolutionary Design and Looking Ahead approach described in this thesis, where the interaction designers were creating an interaction design for a set number of iterations ahead of what the developers were implementing. This is not Parallelisation, however, as the interaction design was progressing along with the system code at every iteration.

The study by Chamberlain, Sharp and Maiden [23] cites projects that included up-front work, such as requirements gathering and understanding users. This agrees with the observations in this thesis, that all teams attempted some of this type of work before implementation began. There is not much further discussion regarding the organisation of activities once implementation had begun, and again, one would have to infer that their five principles for integrating user-centered design and agile development would apply to both Comprehensive Design and Evolutionary Design approaches.

Finally, our findings about the inevitability of up-front design agrees with the study by McInerney and Maurer [93]. Their participants were also able to undertake activities such as contextual inquiries and developing personas prior to coding, in order to drive subsequent design activities. Issues discussed in this thesis, such as the continual involvement of the interaction designer during development in guiding the direction of the interaction design, and involvement of the whole team in interaction

design decisions were identified by McNerney and Maurer’s participants, but not explored further in their study.

9.3 Topics for further discussion

During the course of our analysis for this study, we identified categories that addressed the research questions we set out to investigate, however there also emerged categories that described important phenomena outside the scope of this thesis. We describe two phenomena here as subjects for further discussion.

The first concerns the idea of design — how interaction designers and developers make use of the same word to refer to different concepts — and how these ideas influence the attitude to combining interaction design and agile development. The second concerns the timing of interaction design iterations and agile development iterations, with particular reference to the timing of the feedback from evaluations.

9.3.1 Design: Agile values and cost of change

Throughout our investigation into how teams combined interaction design and agile development, and whether a comprehensive or evolutionary approach was employed, a key issue emerged relating to the idea of *design*. In agile development even the word is regarded with caution. The agile manifesto includes the preferences: “Working software over comprehensive documentation” and “Responding to change over following a plan”, and while these preferences are intended to discourage fixing requirements with plans and documentation, they are often interpreted as a reduction of emphasis on design of any kind [55].

In the literature, much of the discussion relating to up-front design appears in the context of the design of the system code, and yet the aversion to up-front design was seen to be transferred to interaction design on agile

projects as well. P4's project management was one such example. In section 7.5, the participant from P4 is quoted as saying that "it has been quite difficult to put design up front." due to:

[P4.398] "Some people who see themselves as very agile, they think design up front is something reckless." — *Interaction designer, P4*

Even among our participants, we encountered divided opinions regarding the gains of doing design and the cost of doing so before development begins, hence the different approaches to designing and implementing their interaction designs.

A question of nomenclature

Agile development advocates are wary of up-front design because it represents premature commitment. Most emphatically, agile development has typically warned of the dangers of BDUF [2, 53], suggesting that the more the design is determined up front, the more expensive it is to change in the future, when time and experience may make change desirable. Further, when change is inevitable, the time and effort put into creating the big design up front is seen as wasted [9, 87].

Consideration of the place of interaction design in agile development is therefore affected by the nature of interaction design as a kind of design, and the attitude to design within agile development. Interaction design includes the term *design*, and the term has persisted through changes in nomenclature and disciplinary emphasis from user interface design, user interaction design and more recently to user experience design. In fact, interaction designers, including those in our study will sometimes refer to their work as simply "design", implicitly meaning the interaction design — the aspects of the software that the user will see and with which they will interact. At the same time, software developers will use the term "design" to mean the design of the internal structure of the software, the aspects of the software that affect functionality and execution performance

([128] for example). It is important to realise these differences in nomenclature. In particular, it means that agile software developers who have ideas about the places of design tend to apply them to interaction design, even if they have little experience of the interaction design process.

An important consideration, however, is that the up-front interaction design activities observed in our study, involved much close work with business analysts, markets, clients, and end users. Our participants agreed that this work was part of what they regarded as interaction design. On the other hand, this kind of work might also be seen by developers as constituting analysis rather than design. So while this kind of work can be argued as a reason for a large proportion of up-front interaction design, it might also be seen as an argument for up-front analysis.

If up-front design does not imply fixing the requirements up front and if up-front design is analysis and does not involve commitment, then it does not constitute the same kind of danger.

Fowler and design

Martin Fowler asked: “Is Design Dead?” [52]. In considering an answer to this question, he distinguishes *planned design* from *evolutionary design*. In particular, he highlights how iterations and refactoring can mean that evolutionary design can turn out well. He explains:

In its common usage, evolutionary design is a disaster. The design ends up being the aggregation of a bunch of ad-hoc tactical decisions, each of which makes the code harder to alter.

But then goes on to say that the practices of eXtreme Programming make a significant difference:

These enabling practices of continuous integration, testing, and refactoring, provide a new environment that makes evolutionary design plausible.

For some of our participants, following an evolutionary strategy for creating the interaction design worked well, but the process was not perfect and seemed to be one of trial and error. For example, one of the insights that working with an implemented interaction design brought was the over-specifying by the interaction designers of what developers could implement in an iteration (see section 7.3). The challenge appears to be to find the right amount of interaction design up-front that avoids premature commitment, and therefore wastage of resources, but provides the developers with enough guidance to get through the iteration:

[P7.609] “That a certain amount of information has to be known up front ... How much you have to do before you say that’s enough, I think that’s a skill and the experience of working with agile: To say you’ve done enough and feel comfortable about proceeding with this.” — *Developer2, P7*

[P8.655] “I can see that the developers aren’t happy because they’re saying, ‘Well, there’s not enough details.’ ... We’re meeting our objectives about 50% right now, because the interaction designer is not looking at iterations three and four. So he’s not being forward-thinking. He’s not being a visionary ... He’s providing very little detail about iteration three — enough to get about here and not getting it the whole way through [the iteration]. They have to do enough.” — *Project manager, P8*

This suggests a poor understanding of incremental design in terms of the interaction design. The practices of XP allow the incremental design in evolutionary design to be successful, however there is a noticeable lack of guidance in the literature regarding incremental interaction design. Until this concept is better understood, interaction designers on agile teams may have to continue their work by trial and error.

9.3.2 Implemented software vs. unimplemented prototype

While our participants had an understanding that the issues of cost and time determine whether the interaction design should have iterations, and therefore evaluations, with a prototype rather than working software, there was also the observation that interaction design iterations and agile development iterations may have trouble keeping up with one another. One participant admitted that the interaction design iterations often had trouble keeping up with the fast-paced coding iterations:

[P3.328] “With faster and quicker iterations in agile maybe sometimes you need the slower; With the user interface it takes longer to get feedback, so it doesn’t always line up.” — *Developer2, P3*

In the debate organised by Nelson [97], Cooper suggests that the implementation process is too slow and expensive to be used in the iterative process of interaction design and evaluation. In the literature, the view that interaction design iterations are faster than coding iterations, often refers to evaluation with users in the form of an unimplemented prototype.

High vs low fidelity

Usability testing is a fundamental concept of user-centered design. In practice, evaluation of an interaction design involves using both implemented and unimplemented forms. Unimplemented interaction designs may be evaluated in the form of pen and paper or wizard of Oz prototypes during walkthroughs or heuristic evaluations, while evaluation of an implemented interaction design involves evaluating the actual working software, which can be subjected to more formal testing in the field or in the laboratory.

In the literature there is a related debate which authors refer to as the “Low-versus-High-Fidelity-Prototyping” debate [120], where evaluation of implemented and unimplemented forms of the interaction de-

sign are debated with respect to their strengths and weaknesses in terms of the type of feedback they elicit from users. Typically, unimplemented forms of the interaction design are good for eliciting feedback regarding the requirements for the product under development [120]. Pen and paper sketches, for example, can act as an aid for users involved in the evaluations to communicate ideas to the evaluator, that they would otherwise find difficult to verbalise, but does not provide the exact experience of using the actual software [120]. Implemented interaction designs, on the other hand, are better for providing the user with a more accurate sense of what it would be like using the actual product.

The teams in our study evaluated their interaction designs using both unimplemented prototypes and implemented software and considered not only the type of feedback of the form of the interaction design, but also how easy it was to create and change. Participants echoed the reasons cited in the literature for evaluating an interaction design with an unimplemented interaction design, i.e., the low cost of creating and changing designs in this form [34, 120]. Teams who evaluated their product's interaction design using lightweight prototypes, in the form of pen and paper sketches or PowerPoint slides, reasoned that these prototypes were faster to create and easier to change than working software. Having a lightweight prototype made it possible for the teams to go through the design of the whole product with the customer and obtain valuable feedback early on in the development process, before implementation began (see section 5.5).

Teams who evaluated their product's interaction design as implemented software talked about users' (and developers') insights that using actual software brought (see section 7.3). This observation is not surprising as Coyne et al. discuss the ability of different media to reveal and conceal properties of the design [38]. Studies have shown that the higher the fidelity of the prototype used to evaluate a product's interaction design, the more reliably the user's behaviour with respect to the interaction

design can be captured, i.e., the prototype should be “as realistic as possible” [86, 34]. However, evaluating working software with a user is only worthwhile after considerable effort has been invested in development and changing the design once it is implemented is comparatively more costly than changing an unimplemented prototype. One participant explained the difficulty with which an implemented interaction design was made more consistent:

[P1.155] “I’d designed this task manager application and then designed the document manager application and the two things were just so different. So it was like ok, well these are really, really different. I didn’t have any holistic view of all the applications and everything. So, we designed the third application and then ‘Oh wow, I see a lot of commonality now between the three.’ So, we refactored and changed the UI model . . . so now if you look at it, creating a document is very similar to creating a task, which is very similar to creating an issue. So, you know, we had to iterate on those things and it was pretty painful for the engineers. We’re working with a web application and the amount of UI work to get things done precisely as we wanted them is a lot. So, you know, it was just an incredible amount of work to change them. It was like, huge amounts of redoing pages and going through stuff.” — *Product manager/UI designer, P1*

Understanding feedback and change

Not only did we see evidence that implemented interaction designs are difficult to change, there is also evidence that feedback from user evaluations causes the interaction design iterations to be far slower than the agile development iterations.

Formal usability evaluations with working software was found to be the most problematic for the teams, not only in terms of coordinating the testing with the development iterations, but also in terms of accessing the appropriate end users:

[P1.98] “I think I wouldn’t know when, in our process to perform usability testing.” — *Engineering manager, P1*

[P3.254] “The task based testing, we generally try and plan for that and try and fit that in at some stage and that can actually be quite different from the time you do the work — which tends to be problematic because that feedback isn’t coming at the right time. But it’s one of those constraints. The people that you really want to give you feedback also have their own jobs if they didn’t they wouldn’t be particularly useful people for that [testing]. It’s a catch twenty-two” — *Developer, P3*

Whereas the agile workflow for coding — test-code-refactor — can be performed repeatedly in a matter of hours, maintaining an equivalent design-evaluation workflow for the implemented interaction design is problematic. The feedback cycles of an implemented interaction design take longer than the feedback cycles for coding, since organising evaluations with end users takes much preparation and organisation and the customer is not necessarily available continuously throughout the implementation stage of development.

In traditional software engineering processes, unimplemented prototypes of the interaction design are evaluated during the requirements elicitation stage of a development project, while the interaction design of the implemented software is evaluated during usability testing of the final product [21]. The structure of agile development however, differs to the traditional structure, with new requirements emerging continuously throughout the development effort and developers performing continuous testing. Agile development produces software through not only iterative but also incremental development. In section 9.3.1 above, Fowler cites the XP practices as allowing an iterative and incremental approach (evolutionary design).

The teams in our study who followed the Comprehensive Design strategy understood that the UI design process involved iterative development

of prototypes, which were carried out before any implementation began. Usability testing was completed using lightweight prototypes created before development began, and thus did not feature in iterative cycles of development. This approach to interaction design allows only iterative development of the interaction design.

Teams in our study who followed the Evolutionary Design strategy allowed the interaction design to evolve in a more incremental fashion. Usability tests were performed using lightweight prototypes created before implementation — on an interaction design that did not include the entire set of features of the final product — but fitting user evaluations with an implemented interaction design into the structure of agile iterations were problematic. Again, we point out the lack in the literature regarding incremental interaction design, and that even when the term ‘incremental’ is used in an interaction design context, the process described is more akin to iterative interaction design ([41, 66], for example). This creates confusion not only with respect to how evolutionary interaction design work can be governed less by trial and error, but also to the understanding of how to maximise the benefit of interaction design feedback and change within the structure of agile iterations.

9.4 Further work

This thesis has answered the question of how interaction design and agile development are being combined in practice, but time and location constraints limited the research activities that could actually be performed. To build on the results we suggest the following improvements to the study we have undertaken here:

- To collect data at different stages of development on the same projects.
- To collect data from different sources (e.g. participant interviews, in situ observations and team documents) for better triangulation.

- To follow up with end users of the projects to determine and compare user satisfaction.

Further, we suggest that for a better understanding of the most efficient combination of interaction design and agile development, that the issues of the type of interaction design feedback and the cost of change is investigated more fully. In particular, that the forms of interaction design used in evaluations in an agile context be explored and how incremental interaction design can contribute to the understanding of how interaction design and agile development can work together.

Appendix A

Project profiles

Our study is based on interviews relating to real projects, and no two real projects are exactly the same. In our study, for example, some of the projects involved completely new software and new UI designs (P2, P4, P5, P6, P8.2), while others were new versions of existing software and existing UI designs (P1, P3, P7, P8.1, P8.3). Each project had its own story. Here we present some of these stories to highlight the variety of actual practice.

A.1 P1

The first project, P1, involved a new version of existing software. They did not do any overall UI design up front, and their UI designer would have the UI design (HTML mock-up) ready only for the next iteration, sometimes for the next release, designed only for the user stories for that iteration or release. Before the next release (sometimes during the previous release), the user stories and the UI were fleshed out in order to obtain high-level estimates for that release, avoiding these activities from holding up the development process. It was understood that the UI could and should change during development. The development team and UI designer discussed UI issues and their implementation on a daily basis

during development, so both the UI designer and the development team understood how the UI design worked and how it was changing. Overall, the developers had significant input into the UI design. Participants mentioned that the UI could change due to features being cut out and changed during the development process, and so they believed it saved them time to not have a complete UI designed up front. Their UI could also change due to customer feedback or as a result of a usability study. Before an iteration planning meeting, the UI designer and the engineering manager would go through the UI and the user stories it implemented for that iteration. Ideas for the UI design were exchanged at that time. Then the interaction designer would go through the UI and user stories with the whole engineering team, making sure that the developers understood the UI, and again developer feedback and discussions sometimes led to changes to the UI design. Next, in the iteration planning meeting, developers and the UI designer had further discussions about the UI and if the resulting changes were not too major, they could still be incorporated into the UI design. The UI designer also made sure that the UI design was based on how much work it was going to take to implement that design, and that the developers did not have to spend too much time on UI implementation. At this meeting the UI designer, engineering manager and project manager decided on which cards would go into an iteration. Then the cards were implemented by the pair programmers and once they had finished, the cards were ready for acceptance. During acceptance testing the team walked through the UI features manually, checking their usability and how well they worked. The team had the HTML mock-up projected on one screen and the actual application projected on another screen side by side. Large fixes were then written as new user stories for another iteration and smaller fixes were given back to the engineers to fix straight away. The UI was iteratively designed and implemented with an overall goal or metaphor in mind – an idea that was never written down, but was understood by everyone in the team. This idea guided the development

of the UI. It was seen as a waste of time to design too far ahead, as during the development process features were often changed or dropped and the discussions between the UI designer and the rest of the team could lead to changes in the UI.

A.2 P2

The second project, P2, began with the marketing department verifying that the product was viable. Once this had been done, the UI designers started the UI design of the product. XP was seen to begin only after the UI design had been completed, and the UI designers had passed the product on to the developers. According to P2s participants XP began with the development planning meetings. During the first meeting of the process, cards were created from the user stories and prioritized to determine the order of development. Each card took 2 weeks to implement. In this pre-release planning meeting, they also estimated when the product was going to be released. Development of their products usually lasted three to four iterations in total and each iteration lasted 2 weeks. Before each iteration, an iteration planning meeting was held. P2s participants described their implementation process as a “tight development cycle.” Within the iterations, the developers performed unit testing and refactoring every five minutes and created builds hourly. As a component of the product was implemented, all the tests, including the acceptance tests for that component were created at the same time. The UI designers then tested these builds daily. They checked the cards that were implemented the day before, from the perspective of the UI, and if they were happy with them the builds were signed off. If not, feedback regarding changes or omissions were given to the developers. Every few days the domain expert tested the UI too. In both instances large fixes could result in new cards being created, in which case the domain expert, developers and UI designers would negotiate about which iterations would include the new cards. UI issues

were dealt with in the same way as any other development issue was dealt with: for large fixes new cards were created, new price calculations made and all cards were re-prioritized for iterative implementation. However, since the UI was 90% defined before development iterations began, there was little room for changes in the UI from one iteration to the next. Other tests for the UI included a check that heuristics and navigational design rules were met, and usually after the first iteration, the product was tested with the client.

A.3 P3

The third project, P3, also involved a new version of an existing product. They had adopted a set of XP practices to suit their needs, but the process was still not quite where they envisioned it to be. Up front a written requirements specifications document was created for the basis of the sales agreement with the customer and the user stories and estimates were created from that document. However, during development, UI requirements could still change sometimes every three months, sometimes daily. The development team gathered just enough information needed to get started on the UI, e.g., what the UI needed to do and what interactions made up a function, and then they evolved the design from that. The UI design started out as either a pen and paper or PowerPoint prototype, which was then shown to their customers or users for feedback. The prototyping phase sometimes consisted of several iterations of prototyping and obtaining user or customer feedback. Then the prototype was implemented and a concrete piece of software could be shown to the users or customers for feedback. The front end was explained to be well-separated from the back end so that it was easy to switch between different UIs if they changed. The prototyping phase took place in parallel with the coding phase and were not seen to drive development of the system. The requirements specification document formed the basis of the teams long

term plan for the product. This plan was recapped weekly, to determine the teams progress and to decide on the work for the next week. Current bugs, user feedback and other issues with the software would also be discussed on a weekly basis and incorporated into the work for the week. Then the developers went off and did what had been decided on for that week. Interaction design issues were not planned for, it was just seen as part of the work done in an iteration. After the standard XP activities for an iteration had been completed (code, unit test, acceptance test), the UI was run on a simulator and checked. Fixes could result from being run on the simulator (and dealt with as any development fix) or the UI could be left alone for several iterations, without further development or changes. During development, team members designed UI screens and asked fellow team members for feedback as they went, essentially performing expert reviews. Taskbased testing (with users) was more explicitly planned for, but this was performed only when there was enough time. As a result, the feedback from these tests were not coming at the right time for them to be useful or easily incorporated into development, especially when there was a big time lag between the development of the UI and the test. Small fixes resulting from these tests and the expert reviews could be fixed right away, whereas larger fixes were made into new user stories. Sometimes, however, the product was released without the UI being tested at all. The interaction designer saw this as one of the main challenges of incorporating UI design with XP that the UI activities have trouble keeping up with the fast paced coding iterations. Although usability was seen as beneficial, participants admitted that its techniques were traded off against development time. The teams lack of experience was cited as a reason why formal usability techniques were not a part of their process. It was one participants opinion that developers dont need to be the best at solving usability issues, they just need to take into consideration the fact that their products will be used by other people, and that the teams skills and experience would improve over time. The team also had UI guidelines and standards

to adhere to during implementation. At the time of the interviews, P3s main aim was to have working software at all times.

A.4 P4 and P5

P4 and P5, involve two participants who were not working on the same project, but they described very similar processes in use at their organisation. Both participants were involved in Scrum development processes. The UI team designed the UI 90-95% up front, and their UI design was regarded as the specification for their developers. The UI design process was seen as a separate process from the development process and Scrum was seen to kick in after UI design had been completed. They preferred to remove the UI design from the implementation process, as the UI design, and feedback about the design, was seen as coming too late in the process. The UI design was used to create the backlog items and estimates for negotiation with the customer. Once the customer had given their approval, implementation began. Before each iteration, the work for that iteration was planned. Any unfinished features from the previous iteration and the highest priority back-log items became work for that iteration. UI related work during the iteration was request-based — the developer would request help from the UI designer when issues with implementation arose. It was found that some UI decisions could only be made once the internals of the system were known. This meant that some UI design decisions could only be made once implementation had begun. At these times the developers were able to give feedback about UI issues. There was no user involvement during the implementation process, as all user testing took place up front. The UI designers were approached by the developers during the development process to check usability informally. So the UI was occasionally checked (mainly to ensure that the UI matched the requirements specification) in an unsystematic way. If the customer did not think it was necessary, there was no formal usability testing of the fi-

nal product. This was seen as a disadvantage, since relying on developers to approach the UI designers with issues, was found to be unreliable. If developers had forgotten to clear some UI issue up with the UI designers, and that issue happened to affect usability, the product was released to the users with that defect. This meant that users suffered with this defect for several months, before the development team got to correct the problem. At the time of the interviews, the UI designer felt that the release plan was very vague. The release dates were fixed, whereas the features that would be included in that release were not. This meant that features were often shipped in a state that the UI designer was not happy with, who would have preferred the release plan to be tied to completed functionality.

Appendix B

Interview questionnaires

B.1 Initial

B.1.1 Background

- Could you give me a little background about the project you are working on? Can you describe very briefly what it is that your team is building?

Project size and duration?

Roles in the team?

B.1.2 Requirements

- Can you describe the process of gathering the requirements for your system?

Do you collect any requirements especially for the user interface?

Would you say that the requirements change much from one iteration to the next — in terms of the ui and the system as a whole?

B.1.3 Teams

- Which team members are involved in ui/interaction design and how would their roles be described?
- Who owns the ui and/or its code?
- Do you support the notion of having one or several ui/interaction design specialists as part of the team?
- Do you have experience of working in teams including ui/interaction design specialist(s)?
- Would you have this team make-up again in future projects?
- Have you had an XP coach on board to help out the team and what was their contribution to the UI/interaction design side of development?

B.1.4 Process

- Do you experience any bottlenecks during the course of the project due to ui/interaction design issues (such as testing)?
- Can you describe what one iteration of your agile process consists of?
- Does your process incorporate any other cycles?
- How do you incorporate ui/interaction design issues into a release plan (if there is one)?
- What tools do you use for user interface design/implementation and why?
- What ui design tools have you used in the past (agile projects) and why did you change?

- How much up front design do you do and what do you see as the advantages of doing up front design?
- What techniques are used to communicate between the designers and developers?

B.1.5 Usability

- How important are usability issues to your projects?
- Have you set any usability goals for any of the projects? e.g. effective to use, efficient to use, safe to use, have good utility, easy to learn, easy to remember how to use etc.

Were they achieved? (If not, why?)

- Have you set any user experience goals for any of the projects? e.g. satisfying, enjoyable to use, aesthetically pleasing, helpful, motivating, supporting creativity etc.

Were they achieved? (If not, why?)

- What do you do (if anything) during development to ensure usability is built into the software/usability goals are met?
- Has this made a difference to the user satisfaction of software that was developed without much attention to usability practices?
- What techniques do you use during development to take into account the different users of the system?

B.1.6 Testing and refactoring

- Do you do any usability/UI testing?
- How do you integrate usability testing within the development cycle? e.g. how often do you test and when?

- How do you go about planning the fixes that result from the usability/UI testing?
- Would you see releasing the software to the users as a type of usability testing?
- In your experience, would you say that the merciless refactoring of code has a significant effect/not much effect at all on the design of the user interface?
- Would you say that you have refactored the UI?
Do you use specific tools for this job?
- Any other tests you do that involve the user interface?

B.1.7 Users/customers

- Do you follow up with users to determine how satisfied they are with your product?
How do you do that?
- On site customer?
- Is the customer the end user?

B.1.8 Wrap-up

- What has been the biggest challenge with respect to the user interface in developing your system?
- Are there any issues that you think are important, that we haven't discussed?

B.2 Revised

B.2.1 Background

- Could you give me a little background about the project you are working on? Can you describe very briefly what it is that your team is building?

Project size and duration?

Roles in the team? Dedicated interaction design specialist?

On site customer?

Is the customer the end user?

B.2.2 Experience with combining interaction design and agile development

- Where would you say your interaction design process ends and agile development begins? Should there be boundaries?
- Do you think your process works well? Why?
- Where do you see differences between the interaction design process and the agile development process - that might either cause difficulties or enhance the overall development process?

B.2.3 Process

- Are requirements fixed up front? What are the advantages/ disadvantages of this?
- What else happens up front before any development begins?
- What techniques do you use for interaction design?
- How does the implementation process influence the UI?

- When, during the development iteration, does work on the UI happen?
- Where does testing the UI fit into the iteration?

When do you test with users?

How do you go about planning the fixes that result from the usability/UI testing?

- Do interaction design issues result in bottlenecks during the course of development?
- Is up front design (if it is done) enough to ensure usability is built into the software?

How would you describe the user satisfaction of your product?
Why?

- Are there aspects of your development process that have changed over time?

B.2.4 Teams

- Do you consider the code of the UI to be owned by the whole team?
- What are the responsibilities of the developers with respect to the UI?
- How would you describe the communication between developers and designers?

Frequent, daily, etc.

What does that communication mainly consist of?

- Have you had a coach on board to help out the team and what was their contribution to the UI/interaction design side of development?

B.2.5 Wrap-up

- What has been the biggest challenge with respect to combining interaction design and your agile process? Has it been difficult to combine interaction design with your agile development process?
- Are there any issues that you think are important, that we haven't discussed?

Appendix C

Participant information and approved HEC application



HUMAN ETHICS COMMITTEE

Application for Approval of Research Projects

Please email applications to your supervisor who will then email it to an Informatics HEC member for a preliminary review.

Note: The Human Ethics Committee attempts to have all applications approved within three weeks but a longer period may be necessary if applications require substantial revision.

1 NATURE OF PROPOSED RESEARCH:

(a) **Student Research**

(b) If Student Research: Degree: **MSc (part-time)**

(c) Project Title: **User Interface Design in an Agile Environment**

2 INVESTIGATORS:

(a) Principal Investigator

Name **Jennifer Ferreira**

e-mail address: **jennifer@mcs.vuw.ac.nz**

School/Dept/Group **Mathematics, Statistics, and Computer Science**

(b) Other Researchers	Name	Position
	Roger Cliffe	technician

(c) Supervisor (in the case of student research projects)

James Noble	Professor
Robert Biddle	Associate Professor

3 DURATION OF RESEARCH

(a) Proposed starting date for data collection – After HEC approval has been granted.
(Note: that NO part of the research requiring ethical approval may commence prior to approval being given)

(b) Proposed date of completion of project as a whole : **04/04/2007**

4 **PROPOSED SOURCE/S OF FUNDING AND OTHER ETHICAL CONSIDERATIONS**

(a) Sources of funding for the project

Please indicate any ethical issues or conflicts of interest that may arise because of sources of funding
e.g. restrictions on publication of results

None

(b) Is any professional code of ethics to be followed

N

If yes, **name**

(c) Is ethical approval required from any other body

N

If yes, name and indicate when/if approval will be given

5 **DETAILS OF PROJECT**

Briefly Outline:

(a) The objectives of the project

To assist research objectives for MSc thesis and to explore the process and practices of user interface design in an agile environment. Agile software development projects pose special challenges to the design of the user interface. Inherent in agile processes are certain characteristics that conflict with traditional user interface design, especially the fact that no detailed over-arching plan exists for the system under development and changes to requirements are frequent.

In the literature there exist clear-cut rules and recommendations for how to develop and manage the code of a system under agile development and these are well documented. Yet, this is not the case for the user interface. For this reason it would be extremely valuable to attempt to understand the process as it is played out in real projects and document the findings. In previous research on agile methods, interviews with practitioners have proved very successful in helping to understand agile development practices. After an interview, if the participant agrees, an observation session (of the work environment and processes) could be held to reinforce the information gained through an interview.

(b) Method of data collection

These observations will be recorded with the following methods:

- handwritten or typed notes
- voice recorder (if the participant agrees)
- still pictures (if the participant agrees)
- video (if the participant agrees)

Some time after the interview, interviewees will be shown the transcript, to provide them with the opportunity to ensure factual material is recorded accurately. At this point they may withdraw from the study and have their data discarded without question or penalty, if they wish.

(c) The benefits and scientific value of the project

This project should provide valuable insights into how user interfaces are developed as part of an agile software development project. This would also be an opportunity to contribute to the literature on user interface design in an agile environment.

(d) Characteristics of the participants

Members of software development teams that use agile methodologies such as XP, SCRUM etc. during product development. These team members may include developers, user interface designers, technical writers and software architects.

(e) Method of recruitment

Members of software development teams that use agile development methodologies will be asked on a voluntary basis to participate in an interview. These contacts are established during conference attendance or via email and determined by availability and location.

(f) Payments that are to be made/expenses to be reimbursed to participants

Participants will not be paid for taking part in an interview.

(g) Other assistance (e.g. meals, transport) that is to be given to participants

None, unless specifically requested and approved beforehand.

(h) Any special hazards and/or inconvenience (including deception) that participants will encounter

The interviewees will be verbally interviewed in a location that is most convenient to both them and the interviewer (Jennifer Ferreira). No special hazards or inconvenience will be involved beyond those inherent in conversation. No deception will be involved.

If participants agree to an observation session, then video and still images may be taken in order to record the work environment and processes that are observed during the session. This is to aid the memory of the investigator and there will be no special hazards or inconvenience involved beyond those inherent in having a picture taken and being videoed. Again, no deception will be involved.

(i) State whether consent is for: (Please indicate as many as it applies)

- | | |
|---|----------|
| (i) the collection of data | Y |
| (ii) attribution of opinions or information | N |
| (iii) release of data to others | N |
| (iv) use for a conference report or a publication | Y |
| (v) use for some particular purpose (specify) | Y |

The consent is for the collection of data. This data will be analysed and presented in the investigator's research project. Publications and reports may also be produced as a result of this study and participants will be notified of any report or publication that results, if they wish to know.

Attach a copy of any questionnaire or interview schedule to the application

Y

(j) How is informed consent to be obtained (see paragraphs 4.31(g), 5.2, 5.5 and 5.61 of the Guidelines)

(i) the research is strictly anonymous, an information sheet is supplied and informed consent is implied by voluntary participation in filling out a questionnaire for example (include a copy of the information sheet)

N

(ii) the research is not anonymous but is confidential and informed consent will be obtained through a signed consent form (include a copy of the consent form and information sheet)

Y

(iii) the research is neither anonymous nor confidential and informed consent will be obtained through a signed consent form (include a copy of the consent form and information sheet)

N

(iv) informed consent will be obtained by some other method (please specify and provide details)

N

The individual identity of an interviewee or observed person is not relevant to the research, so no personal information will be collected. Only the investigator and the supervisors will have access to the original notes that allow participants to be individually identified. In any report or publication, individuals will be referred to by number, code, or pseudonym. Original notes will be kept in secure storage at the university, and destroyed one to two years after the completion of the project.

With the exception of anonymous research as in (i), if it is proposed that written consent will not be obtained, please explain why

(k) If the research will not be conducted on a strictly anonymous basis state how issues of confidentiality of participants are to be ensured if this is intended. (See paragraph 4.3.1(e) of the Guidelines). (e.g. who will listen to tapes, see questionnaires or have access to data). Please ensure that you distinguish clearly between anonymity and confidentiality. Indicate which of these are applicable.

(i) access to the research data will be restricted to the investigator

N

(ii) access to the research data will be restricted to the investigator and their supervisor (student research)

Y

(iii) all opinions and data will be reported in aggregated form in such a way that individual persons or organisations are not identifiable

Y

(iv) Other (please specify)

Only the investigator, technician and the supervisors will have access to the data.

- (l) Procedure for the storage of, access to and disposal of data, both during and at the conclusion of the research. (see section 7 of the guidelines). Indicate which are applicable:
- (i) all written material (questionnaires, interview notes, etc) will be kept in a locked file and access is restricted to the investigator **Y**
 - (ii) all electronic information will be kept in a password-protected file and access will be restricted to the investigator **Y**
 - (iii) all questionnaires, interview notes and similar materials will be destroyed:
 - (a) at the conclusion of the research **N**
 - or (b) 2 years after the conclusion of the research **Y**
 - (iv) any audio or video recordings will be returned to participants and/or electronically wiped **Y**
 - (v) other procedures (please specify):

If data and material are not to be destroyed please indicate why and the procedures envisaged for ongoing storage and security

Any original notes or recordings will be kept in a locked file and will be accessible only to the investigator and the supervisors. These notes or recordings will be destroyed by shredding (notes) and deletion (computer files, video and voice recordings) at the end of the project.

- (m) Feedback procedures (See section 8 of the Guidelines). You should indicate whether feedback will be provided to participants and in what form. If feedback will not be given, indicate the reasons why.

Participants will be notified of the availability of the electronic form of the thesis on the web via email, as well as of any report or publication based on their participation, if they wish. The thesis will be available at the conclusion of the principal investigator's MSc. A prospective date for this is April 2007. The first publication will be made available in June 2006.

Some time after the interview, interviewees will be shown the transcript, to provide them with the opportunity to ensure factual material is recorded accurately. At this point they may withdraw from the study and have their data discarded without question or penalty, if they wish.

- (n) Reporting and publication of results. Please indicate which of the following are appropriate. The proposed form of publications should be indicated on the information sheet and/or consent form.
- (i) publication in academic or professional journals **Y**

- | | |
|--|----------|
| (ii) dissemination at academic or professional conferences | Y |
| (iii) deposit of the research paper or thesis in the University Library (student research) | Y |
| (iv) a case study used for teaching purposes | Y |
| (v) other (please specify) | N |

Signature of investigators as listed on page 1 (including supervisors) and Chair of Informatics HEC.

NB: All investigators and the Chair of Informatics HEC must sign the form, then send it to Perumal Pillai for filing in the University's Research Office once the electronic application has been approved.

..... Date.....

supervisors:

..... Date.....

..... Date.....

Chair of Informatics HEC:

..... Date

INFORMATICS APPLICATIONS FOR HUMAN ETHICS APPROVAL

CHECKLIST

- ✓ Have you read the Human Ethics Committee Policy?
- ✓ Have you read the Informatics HEC Guide?
- ✗ Is ethical approval required for your project?
- ✗ Have you established whether informed consent needs to be obtained for your project?
- ✗ In the case of student projects, have you consulted your supervisor about any human ethics implications of your research?
- ✓ Have you included an information sheet for participants which explains the nature and purpose of your research, the proposed use of the material collected, who will have access to it, whether the data will be kept confidential to you, how anonymity or confidentiality is to be guaranteed?
- ✓ Have you included a written consent form?
- ✓ If not, have you explained on the application form why you do not need to get written consent?
 - Are you asking participants to give consent to:
 - ✓ collect data from them
 - ✗ attribute information to them
 - ✗ release that information to others
 - ✓ use the data for particular purposes
- ✓ Have you indicated clearly to participants on the information sheet and/or consent form how they will be able to get feedback on the research from you (e.g. they may tick a box on the consent form indicating that they would like to be sent a summary), and how the data will be stored or disposed of at the conclusion of the research?
- ✓ Have you included a copy of any questionnaire or interview checklist you propose using?

POINTERS TO AVOID HAVING APPLICATIONS RETURNED BEFORE HEC REVIEW

- ▶ The approval process is speeded up by not requiring the hard copy of your application form with the signatures on it at the initial review process. The complete application (HEC application form, info sheet, consent form, covering letter, questionnaire etc.) is to be emailed as an attachment in one file to your supervisor who will email it to an INFORMATICS HEC member for a preliminary review.
- ▶ Do not insert a date into item 3 a.
- ▶ Delete the "Y" or "N" option that is not required. **DO NOT** remove any other text from the application form.
 - ▶ **BOLD** your answers if you wish but do not alter the font anywhere else in the form.



User Interface Design in an Agile Environment: Consent Form

I consent to take part in an interview/observation session about **User Interface Design in an Agile Environment**.

I agree to let Jennifer Ferreira use the data collected to assist her research objectives for her Masters degree, subject to the conditions below:

- I have been fully informed of the purpose and methods to be applied during this interview/observation session,
- I have had the opportunity to ask questions and have them answered to my satisfaction,
- My participation will remain confidential,
- Data presented or published will be stripped of my identity as well as any identifying information,
- I retain the right to withdraw, without question, up until before data analysis begins, two weeks after I receive the interview transcript.

NAME: _____

SIGNATURE: _____

DATE:-----

I agree to have the interview tape-recorded YES / NO

I agree to being observed in my work environment YES / NO

If YES:

I agree to have still pictures taken YES / NO

I agree to having the investigator video my work environment YES / NO

I agree to further discussion at a time other than the time of this interview, if there is any material that is important to the principal investigator's research falling outside the scope of the interview agenda YES / NO

I would like to receive feedback regarding any reports/publications that result from this study YES / NO

If YES: EMAIL: _____

Thank you for your cooperation. If you have any further questions, please contact either of the following:

Principal researcher

Jennifer Ferreira (Student)

School of Mathematics, Statistics and Computer Science, jennifer@mcs.vuw.ac.nz

Supervisor

Prof. James Noble

School of Mathematics, Statistics and Computer Science, kjx@mcs.vuw.ac.nz, +64 4 463 6736



User Interface Design in an Agile Environment: Information Sheet

I am a Master of Science student at Victoria University of Wellington, researching issues in user interface design in an agile environment. As part of my research I would like to conduct an interview/observation session with you, an agile practitioner, in order to learn more about the way user interfaces are developed on agile projects. I value your experiences, opinions and insights into this topic.

Conditions

This interview is conducted by me, Jennifer Ferreira, under the following conditions

- You are fully informed of the purpose and methods to be applied during this interview/observation session,
- You have the opportunity to ask questions and have them answered to your satisfaction,
- Your participation will remain confidential. Any written or electronic material will be stored securely and will not be retained after the completion of the research project,
- Data presented or published will be stripped of your identity as well as any identifying information,
- You have the right to withdraw, without question, up until before data analysis begins, two weeks after you receive the interview transcript,
- You have indicated whether you will allow the interview to be tape-recorded or not,
- You have indicated whether you will allow an observation session to take place or not and whether or not still pictures and video footage may be taken during this observation session,
- You have indicated whether or not you would like to receive feedback of any publications or reports that result from this study.

There will be an opportunity to review any written notes that result from the interview/observation session to ensure factual material is recorded accurately.

Prior to conducting the interview, Victoria University of Wellington require that I obtain your written informed consent. This consent is a normal part of any research project and forms one criterion of the Human Ethics Committee guidelines that I must meet. Therefore you will be provided with a consent form to sign. You will also be provided with an interview agenda.

Scope

An interview agenda has been attached to this information sheet, outlining what information will be sought during the interview. If there is any material that you would like to communicate to the interviewer that falls outside the scope of the interview agenda, the interviewer will make a note in a separate register for further discussion at another time. No personal details or information will be collected in either the interview or the observation session.

Duration and Feedback

The interview will take at most one hour. If you so wish, feedback regarding this study will be made available to you via electronic mail, in which you will be notified of the availability of any reports or publications.

The observation session will last the amount of time that you, the participant will allow. You may end the observation session without question at such a time as you feel is reasonable.

Structure of Discussion

Before commencing the interview you will be provided with this information sheet, a consent form and an interview agenda. If you consent in writing to the interview as described on this information sheet and accompanying interview agenda, then the interview will begin. The discussion will be voice recorded if you consent to it. The interview will not last longer than 60 minutes but as per the consent sheet, you may terminate the discussion at any point during the interview, without question. The transcription of the interview and subsequent findings will be provided to you to ensure all information has been recorded and interpreted accurately. The interview agenda provides the scope of the interview questions but any discussion concerning development practices that fall outside the scope of the interview agenda will be noted in a register for further discussion at another time, if you agree.

Contacts

Thank you for your cooperation. If you have any further questions, please contact Jennifer Ferreira or James Noble:

Principal researcher

Jennifer Ferreira (Student)

School of Mathematics, Statistics and Computer Science, jennifer@mcs.vuw.ac.nz

Supervisor

Prof. James Noble

School of Mathematics, Statistics and Computer Science, kix@mcs.vuw.ac.nz, +64 4 463 6736



User Interface Design in an Agile Environment: Interview Agenda

Interview Details

Date: To be decided

Topic: User interface design in practice on an agile development project

Outcome

- To learn more about the process of developing the user interface as part of an agile project.
- To note the experiences and opinions of an agile practitioner about the practices that currently exist.

Agenda

No.	Description	Duration
1.	Agree outcome, agenda and rules for the interview	5 minutes
2.	Current project involvement	5 minutes
3.	Requirements, teams, process/tools,	20 minutes
4.	Usability, testing, refactoring, follow-up	25 minutes
6.	Wrap-up <ul style="list-style-type: none"> - review progress - next steps 	5 minutes
Total		1 hour

Rules

- The interview will be tape recorded, if the interviewee gives their your approval, to reduce the risk of the interviewer not obtaining all of the information provided by the interviewee during the interview
- The transcription of the interview and subsequent findings will be provided to the interviewee to ensure all information has been recorded and interpreted accurately.
- To ensure all items are covered in sufficient detail, any discussion concerning development practices outside the scope of the interview agenda will be noted in a register for further discussion at another time, if the participant agrees.



User Interface Design in an Agile Environment: Interview Questions

Background/Current Project Involvement

1. Could you give me a little background about the project you are working on? Can you describe very briefly what it is that your team is building?

1.1. Project size and duration?

Requirements

2. Where do you get the requirements for the user interface from?

2.1 Would you say that user stories can provide enough information for user interface design?

2.2 Do you use any other techniques to get requirements?

Teams

3. Which team members are involved in ui design and how would their roles be described?

4. Do you support the notion of having one or several ui specialists as part of the team?

5. Do you have experience of working in teams including ui specialists(s)?

6. Would you have this team make-up again in future projects?

Process and Tools

7. Do you experience any bottlenecks during the course of the project due to ui issues (such as testing)?

8. Can you describe what one iteration of your agile process consists of?

9. Does your process incorporate any other cycles?

10. How do you incorporate ui issues into a release plan (if there is one)?

11. What tools do you use for user interface design and why?

12. What ui design tools have you used in the past (agile projects) and why did you change?

13. XP warns against doing up-front design. In your opinion, does this impair user interface design?

13.1. If your method is not XP, do you do any up-front design - particularly for the user interface?

Usability

14. Some say the reason why there is so little in the literature about agile usability is that usability is not important to XP projects, or the developers are ignoring it.

Can you Comment?

15. Have you set any usability goals for any of the projects?

15.1 Were they achieved? (If not, why?)

16. What do you do (if anything) during development to ensure usability is built into the software/usability goals are met?

16.1 Has this made a difference to the user satisfaction of software that was developed without much attention to usability practices?

Testing and Refactoring

17. Do you do any usability testing?

17.1 What type?

18. How do you go about planning the fixes that result from the usability testing?

19. Would you see releasing the software to the users as a type of usability testing?

20. Do you do/have you done unit testing of the user interface?

20.1 Do you unit test both appearance and behaviour or only one of appearance and behaviour?

20.2 How did you do this?

21. In your experience, would you say that the merciless refactoring of code has a significant effect/not much effect at all on the design of the user interface?

22. Is refactoring the user interface important to you?

22.1. Do you use specific tools for this job?

Follow-up with users/customers

23. Do you follow up with users to determine how satisfied they are with your product?

23.1. How do you do that?

24. Co-located customer?

25. Is the customer the end user?

Wrap-Up

26. How do you see the future of UI design in agile environments?

27. Are there any issues that you think are important, that we haven't discussed?

Bibliography

- [1] ABRAHAMSSON, P., WARSTA, J., SIPONEN, M., AND RONKAINEN, J. New directions on agile methods: A comparative analysis. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, USA, May 3–10* (Washington, DC, USA, 2003), IEEE Computer Society, pp. 244–254.
- [2] AMBLER, S. Quality in an agile world. *Software Quality Professional* 7, 4 (2005), 34–40.
- [3] AMBLER, S. Survey says: Agile works in practice. *Dr. Dobb's Journal* (August 2006). Available at <http://www.ddj.com/architect/191800169?pgno=1>. Last accessed 7 Nov 2007.
- [4] ARMITAGE, J. Are agile methods good for design? *interactions* 11, 1 (2004), 14–23.
- [5] ARUNACHALAM, V., AND SASSO, W. Cognitive processes in program comprehension: An empirical analysis in the context of software reengineering. *Systems and Software* 34 (1996), 177–189.
- [6] AUER, K., AND MILLER, R. *Extreme Programming Applied: Playing To Win*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [7] BECK, K. Extreme programming: A humanistic discipline of software development. In *FASE '98: Proceedings of the 1st International*

- Conference on Fundamental Approaches to Software Engineering, Lisbon, Portugal, March 28–April 4 (1998)*, E. Astesiano, Ed., vol. 1382 of *Lecture Notes in Computer Science*, Springer, pp. 1–6.
- [8] BECK, K. Embracing change with extreme programming. *Computer* 32, 10 (1999), 70–77.
- [9] BECK, K. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, Reading, MA, USA, 1999.
- [10] BECK, K., AND ANDRES, C. *Extreme Programming Explained: Embrace Change*, 2 ed. Addison-Wesley, Boston, MA, USA, 2004.
- [11] BECK, K., AND FOWLER, M. *Planning Extreme Programming*. Addison-Wesley, Boston, MA, USA, 2001.
- [12] BEEDLE, M., DEVOS, M., SHARON, Y., SCHWABER, K., AND SUTHERLAND, J. Scrum: An extension pattern language for hyperproductive software development. In *Pattern Languages of Program Design 4*, N. Harrison, B. Foote, and H. Rohnert, Eds. Addison-Wesley, 1999.
- [13] BEYER, H., HOLTZBLATT, K., AND BAKER, L. An agile customer-centered method: Rapid contextual design. In *XP/Agile Universe '04: Extreme Programming and Agile Methods, Calgary, Alberta, Canada, August 15–18 (2004)*, C. Zannier, H. Erdogmus, and L. Lindstrom, Eds., vol. 3134 of *Lecture Notes in Computer Science*, Springer, pp. 50–59.
- [14] BOEHM, B. W. *Software Engineering Economics*. Prentice Hall, Englewood Cliffs, NJ, USA, 1981.
- [15] BOEHM, B. W. A spiral model of software development and enhancement. *Computer* 21, 5 (1988), 61–72.

- [16] BOEHM, B. W. Some future trends and implications for systems and software engineering processes. *Systems Engineering* 9, 1 (2006), 1–19.
- [17] BOLLES, G. A., AND KIRKPATRICK, T. A. Research, E-Business. *CIO Insight* (December 2001). Available at <http://www.cioinsight.com/article2/0,1540,2325,00.asp>. Last accessed on 7 Nov 2007.
- [18] BRAITHWAITE, K., AND JOYCE, T. XP expanded: Distributed extreme programming. In *XP '05: Proceedings of the 6th International Conference on Extreme Programming and Agile Processes in Software Engineering, Sheffield, UK, June 18–23 (2005)*, H. Baumeister, M. Marchesi, and M. Holcombe, Eds., vol. 3556 of *Lecture Notes in Computer Science*, Springer, pp. 180–188.
- [19] BROOKS, F. P. *The Mythical Man-Month*. Addison-Wesley, 1975.
- [20] BROOKS, F. P. No silver bullet: Essence and accidents of software engineering. *Computer* 20, 4 (1987), 10–19.
- [21] BRUEGGE, B., AND DUTOIT, A. H. *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*. Prentice Hall, Upper Saddle River, NJ, USA, 2000.
- [22] CARROLL, J. M. *Scenario-Based Design*. Wiley, New York, NY, USA, 1995.
- [23] CHAMBERLAIN, S., SHARP, H., AND MAIDEN, N. A. M. Towards a framework for integrating agile development and user-centred design. In *XP '06: Proceedings of the 7th International Conference on Extreme Programming and Agile Processes in Software Engineering, Oulu, Finland, June 17–22 (2006)*, pp. 143–153.

- [24] CHARMAZ, K. Grounded theory. In *Rethinking Methods in Psychology*, J. A. Smith, R. Harr, and L. V. Langenhove, Eds. Sage Publications Inc., London, UK, 1995, pp. 27–49.
- [25] CHARMAZ, K. Grounded theory: Objectivist and constructivist methods. In *Handbook of Qualitative Research*, N. K. Denzin and Y. S. Lincoln, Eds., 2nd ed. Sage, Thousand Oaks, CA, USA, 2000, pp. 509–535.
- [26] CLEGG, C., AXTELL, C., DAMODARAN, L., FARBY, B., HULL, R., LLOYD-JONES, R., NICHOLLS, J., SELL, R., TOMLINSON, C., AINGER, A., AND STEWART, T. The performance of information technology and the role of human and organizational factors. Organisational Aspects of Information Technology Special Interest Group. University of Sheffield, 1996.
- [27] CLOKE, G. Get your agile freak on!: Agile adoption at Yahoo! Music. In *Agile '07: Proceedings of the AGILE 2007 Conference, Washington, DC, USA, August 13–17 (2007)*, IEEE Computer Society, pp. 240–248.
- [28] COCKBURN, A. *Agile Software Development*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [29] COCKBURN, A. Learning from agile software development — Part two. *CrossTalk: Journal of Defense Software Engineering* (November 2002), 9–12.
- [30] COCKBURN, A. *People and Methodologies in Software Development*. PhD thesis, Faculty of Mathematics and Natural Sciences, University of Oslo, Norway, February 2003.
- [31] COCKBURN, A. *Crystal Clear: A Human-Powered Methodology for Small Teams*. Addison-Wesley Professional, 2004.

- [32] COMPUTER AID, INC. A CAI state of the practice interview with Tom Gilb. *IT Metrics and Productivity Institute: Covering Best Practices in Software Development, Management and Maintenance*. Published online at <http://www.itmpi.org/default.aspx?pageid=290>. Last accessed on 7 Nov 2007, 2005.
- [33] CONSTANTINE, L. L. Process agility and software usability: Toward lightweight usage-centered design. *Information Age* 8, 2 (2001).
- [34] CONSTANTINE, L. L., AND LOCKWOOD, L. A. D. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. ACM Press, 1999.
- [35] COOPER, A. *About Face: The Essentials of User Interface Design*. IDG Books, Boston, MA, USA, 1995.
- [36] COOPER, A. *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity*. SAMS, 1999.
- [37] COOPER, A., REIMANN, R., REIMANN, R. M., AND DUBBERLY, H. *About Face 2.0: The Essentials of Interaction Design*. John Wiley & Sons, Inc., New York, NY, USA, 2003.
- [38] COYNE, R., PARK, H., AND WISZNIEWSKI, D. Design devices: Digital drawing and the pursuit of difference. *Design Studies* 23, 3 (2002), 263–286.
- [39] CRESWELL, J. W. *Research Design: Qualitative and Quantitative Approaches*. Sage Publications Inc., Thousand Oaks, CA, USA, 1994.
- [40] CRITTENDEN, K. S., AND HILL, R. J. Coding reliability and validity of interview data. *American Sociological Review* 36 (December 1971), 1070–1080.

- [41] DA SILVA, P. P. User interface declarative models and development environments: A survey. In *DSV-IS '00: Proceedings of the 7th International Workshop on Design, Specification and Verification of Interactive Systems, Limerick, Ireland, June 5–6 (2000)*, P. A. Palanque and F. Paternò, Eds., vol. 1946 of *Lecture Notes in Computer Science*, Springer, pp. 207–226.
- [42] DERBY, E., LARSEN, D., AND SCHWABER, K. *Agile Retrospectives: Making Good Teams Great*. Pragmatic Bookshelf, 2006.
- [43] DIX, A., FINLAY, J., ABOWD, G., AND BEALE, R. *Human-Computer Interaction*, 2nd ed. Prentice Hall, 1998.
- [44] DUVALL, L. A study of software management: The state of practice in the United States and Japan. *Journal of Systems and Software* 31, 2 (1995), 109–124.
- [45] EARTHY, J., JONES, B. S., AND BEVAN, N. The improvement of human-centred processes — facing the challenge and reaping the benefit of ISO 13407. *Int. J. Hum.-Comput. Stud.* 55, 4 (2001), 553–585.
- [46] EISENHARDT, K. M. Building theories from case study research. *Academy of Management. The Academy of Management Review* 14, 4 (1989), 532–550.
- [47] FERRE, X., JURISTO, N., WINDL, H., AND CONSTANTINE, L. Usability basics for software developers. *Software, IEEE* 18, 1 (Jan/Feb 2001), 22–29.
- [48] FERREIRA, J., NOBLE, J., AND BIDDLE, R. Agile development iterations and UI design. In *Agile '07: Proceedings of the AGILE 2007 Conference, Washington, DC, USA, August 13–17 (2007)*, IEEE Computer Society, pp. 50–58.

- [49] FERREIRA, J., NOBLE, J., AND BIDDLE, R. Interaction designers on extreme programming teams: Case studies from the real world. In *NZCSRSC '07: Proceedings of the 5th New Zealand Computer Science Research Student Conference, Hamilton, New Zealand, April 10–13 (2007)*.
- [50] FERREIRA, J., NOBLE, J., AND BIDDLE, R. Up-front interaction design in agile development. In *XP '07: Proceedings of the 8th International Conference on eXtreme Programming and Agile Processes in Software Engineering, Como, Italy, June 18–22 (2007)*, G. Concas, E. Damiani, M. Scotto, and G. Succi, Eds., vol. 4536 of *Lecture Notes in Computer Science*, Springer, pp. 9–16.
- [51] FONTANA, A., AND FREY, J. Interviewing: The art of science. In *Handbook of Qualitative Research*, N. K. Denzin and Y. S. Lincoln, Eds. Sage, Thousand Oaks, CA, USA, 1994, pp. 361–376.
- [52] FOWLER, M. Is design dead? In *Extreme Programming Examined*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001, pp. 3–17.
- [53] FOWLER, M. Put your process on a diet. *Dr. Dobbs's Journal* (June 2001). Available at <http://www.ddj.com/dept/architect/184414675>. Last accessed on 7 Nov 2007.
- [54] FOWLER, M. The New Methodology. Published online at <http://www.martinfowler.com/articles/newMethodology.html>, Last accessed on 4 Jul 2007.
- [55] FOWLER, M., AND HIGHSMITH, J. The agile manifesto. *Software Development* (August 2001), 28–32.
- [56] FUGGETTA, A. Software Process: A Roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland, June 4–11 (New York, NY, USA, 2000)*, ACM Press, pp. 25–34.

- [57] GARRETT, J. J. *The Elements of User Experience: User-Centered Design for the Web*. New Riders Publishers, Indianapolis, IN, USA, 2002.
- [58] GARTNER, L. The rookie primer. Radcliffe Rugby Football Club. Available at http://www.hcs.harvard.edu/~radrugby/rookie_primer.html. Last accessed on 7 Nov 2007, 1996.
- [59] GHEZZI, C., JAZAYERI, M., AND MANDRIOLI, D. *Fundamentals of Software Engineering*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.
- [60] GILB, T. *Principles of Software Engineering Management*. Addison-Wesley, Wokingham, UK, 1988.
- [61] GLASER, B. *Theoretical Sensitivity: Advances in the methodology of Grounded Theory*. Sociology Press, Mill Valley, CA, USA, 1978.
- [62] GLASER, B. *Emergence vs. Forcing: Basics of Grounded Theory Analysis*. Sociology Press, Mill Valley, CA, USA, 1992.
- [63] GLASER, B. Conceptualization: On theory and theorizing using grounded theory. *International Journal of Qualitative Methods* 1, 2 (2002).
- [64] GLASER, B., AND STRAUSS, A. *The discovery of grounded theory: Strategies for qualitative research*. Aldine Atherton, NY, US, 1967.
- [65] GLASS, R. Elementary level discussion of compiler/interpreter writing. *ACM Computing Surveys* (March 1969), 64–68.
- [66] GROSS, M. D., AND DO, E. Y.-L. Ambiguous intentions: A paper-like interface for creative design. In *UIST '96: Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, Seattle, WA, USA, November 6–8 (New York, NY, USA, 1996), ACM, pp. 183–192.

- [67] GROSSMAN, F., BERGIN, J., LEIP, D., MERRITT, S., AND GOTEL, O. One XP experience: Introducing agile (XP) software development into a culture that is willing but not ready. In *CASCON '04: Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative Research, Markham, Ontario, Canada, October 5–7 (2004)*, IBM Press, pp. 242–254.
- [68] HANSEN, B. H., AND KAUTZ, K. Grounded theory applied — studying information systems development methodologies in practice. In *HICSS '05: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA, January 3–6 (2005)*, IEEE Computer Society, p. 264.2.
- [69] HÄTINEN, A. J. Extreme programming and goal oriented user interface design in practice. Research Seminar on Software Engineering. Available at <http://www.pharazon.org/publications/GO-XP.pdf>. Last accessed on 7 Nov 2007, 2002.
- [70] HIGHSMITH, J. *Agile Software Development Ecosystems*. Pearson Education, 2002.
- [71] HIGHSMITH, J., AND COCKBURN, A. Agile software development: The business of innovation. *Computer* 34, 9 (2001), 120–127.
- [72] HODGETTS, P., AND PHILLIPS, D. Extreme adoption experiences of a B2B start up. In *Extreme Programming Perspectives*, M. Marchesi, G. Succi, D. Wells, and L. Williams, Eds. Addison Wesley Professional, 2002.
- [73] ISO TC 159/SC 4; ISO Standards ICS: 13.180. *Human-centred design processes for interactive systems*. International Organization for Standardization, Geneva, Switzerland, 1999.
- [74] KAPLAN, B., AND MAXWELL, J. Qualitative research methods for evaluating computer information systems. In *Evaluating Health Care*

- Information Systems: Methods and Applications*, J. Anderson, C. Aydin, and S. Jay, Eds. Sage, Thousand Oaks, CA, USA, 1994, pp. 45–68.
- [75] KARLSTRÖM, D. Introducing Extreme Programming: An Experience Report. In *XP '02: Proceedings of the 3rd International Conference on Extreme Programming and Agile Processes in Software Engineering, Sardinia, Italy, May 26–29 (2002)*, Springer, pp. 24–29.
- [76] KHALIFA, M., AND VERNER, J. M. Drivers for software development method usage. *Engineering Management, IEEE Transactions on* 47, 3 (Aug 2000), 360–369.
- [77] KNIBERG, H. Scrum and XP from the Trenches. Published online at <http://www.lulu.com/content/899349>. Last accessed on 7 Nov 2007, 2007.
- [78] KRUCHTEN, P. Agility with the RUP. *Cutter IT Journal* 14, 12 (2001), 27–33.
- [79] LAN, C., MOHAN, K., PENG, X., AND RAMESH, B. How extreme does extreme programming have to be? Adapting XP practices to large-scale projects. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA, January 5–8 (Washington, DC, USA, 2004)*, IEEE Computer Society, p. 30083.3.
- [80] LARMAN, C., AND BASILI, V. R. Iterative and incremental development: A brief history. *Computer* 36, 6 (2003), 47–56.
- [81] LARSEN, D. Agile Alliance survey: Are we there yet? *InfoQ* (September 2006). Available at <http://www.infoq.com/articles/agile-alliance-survey-2006>. Last accessed on 7 Nov 2007.
- [82] LAYMAN, L., WILLIAMS, L., AND CUNNINGHAM, L. Exploring extreme programming in context: An industrial case study. In *ADC*

- '04: *Proceedings of the 2004 Agile Development Conference, Salt Lake City, UT, USA, June 22–26* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 32–41.
- [83] LENZ, M., BENO, J., BURNS, M., AND MEANEY, S. Unifying the cisco intranet through hierarchical navigation. In *CHI Extended Abstracts* (2005), G. C. van der Veer and C. Gale, Eds., ACM, pp. 1004–1021.
- [84] LETOVSKY, S. Cognitive processes in program comprehension. In *Papers presented at the first workshop on empirical studies of programmers on empirical studies of programmers* (Norwood, NJ, USA, 1986), Ablex Publishing Corp., pp. 58–79.
- [85] LI, M., BOEHM, B., AND OSTERWEIL, L. Unifying the software process spectrum. *Journal of Software* 17, 4 (2006), 649–657.
- [86] LIFE, A., SALTER, I., TEMEM, J., BERNARD, F., ROSSET, S., BENNACEF, S., AND LAMEL, L. Data collection for the mask kiosk: Woz vs. prototype system. In *ICSLP '96: Proceedings of the 4th International Conference on Spoken Language Processing, Philadelphia, PA, USA, October 3–6* (1996), vol. 3, pp. 1672–1675.
- [87] LINDVALL, M., BASILI, V., BOEHM, B., COSTA, P., DANGLE, K., SHULL, F., TESORIERO, R., WILLIAMS, L., AND ZELKOWITZ, M. Empirical findings in agile methods. In *XP/Agile Universe '02: Proceedings of the Second XP Universe and First Agile Universe Conference, Chicago, IL, USA, August 4–7* (London, UK, 2002), Springer-Verlag, pp. 197–207.
- [88] LINDVALL, M., MUTHIG, D., DAGNINO, A., WALLIN, C., STUPERICH, M., KIEFER, D., MAY, J., AND KAHKONEN, T. Agile software development in large organizations. *Computer* 37, 12 (December 2004), 26–34.

- [89] LINGS, B., AND LUNDELL, B. On the adaptation of grounded theory procedures: Insights from the evolution of the 2G. *Information Technology and People* 18, 3 (2005), 196–211.
- [90] MARSHALL, C., AND ROSSMAN, G. B. *Designing Qualitative Research*. Sage Publications Inc., 1989.
- [91] MARTIN, A., BIDDLE, R., AND NOBLE, J. The XP customer role in practice: Three studies. In *ADC '04: Proceedings of the 2004 Agile Development Conference, Salt Lake City, UT, USA, June 22–26* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 42–54.
- [92] MARTIN, P. Y., AND TURNER, B. A. Grounded theory and organisational research. *Journal of Applied Behavioural Science* 22 (1986), 141–157.
- [93] MCINERNEY, P., AND MAURER, F. UCD in agile projects: Dream team or odd couple? *interactions* 12, 6 (2005), 19–23.
- [94] MICHAEL, D. M. Qualitative research in information systems. *MISQ Discovery* (June 1997).
- [95] MILES, M., AND HUBERMAN, A. *Qualitative Data Analysis*, 2nd ed. Sage Publications Inc., 1994.
- [96] MILLER, L. Case study of customer input for a successful product. In *ADC '05: Proceedings of the 2005 Agile Development Conference, Denver, CO, USA, July 24–29* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 225–234.
- [97] NELSON, E. Extreme programming vs. interaction design. FTP Online. Available at http://www.ftponline.com/interviews/beck_cooper/. Last accessed on 7 Nov 2007, 2002.
- [98] NIELSEN, J. Enhancing the explanatory power of usability heuristics. In *CHI '94: Proceedings of the SIGCHI conference on Human factors*

- in computing systems, Boston, MA, USA, April 24–28* (New York, NY, USA, 1994), ACM Press, pp. 152–158.
- [99] NIELSEN, J. Guerilla HCI: Using discount usability engineering to penetrate the intimidation barrier. In *Cost-Justifying Usability*, R. G. Bias and D. J. Mayhew, Eds. Academic Press, Boston, MA, USA, 1994, pp. 245–272.
- [100] NIELSEN, J. Heuristic Evaluation. In *Usability Inspection Methods*, J. Nielsen and R. L. Mack, Eds. John Wiley & Sons Inc., 1994.
- [101] NIELSEN, J. Return on investment for usability. *Jakob Nielsen's Alertbox* (7 January 2003). Available at <http://www.useit.com/alertbox/20030107.html>. Last accessed on 7 Nov 2007.
- [102] NIELSEN, J. Usability 101: Introduction to usability. *Jakob Nielsen's Alertbox* (25 August 2003). Available at <http://www.useit.com/alertbox/20030825.html>. Last accessed on 7 Nov 2007.
- [103] NIELSEN, J., AND MCMUNN, D. The Agile Journey: Adopting XP in a Large Financial Services Organization. In *XP '05: Proceedings of the 6th International Conference on Extreme Programming and Agile Processes in Software Engineering, Sheffield, UK, June 18–23 (2005)*, H. Baumeister, M. Marchesi, and M. Holcombe, Eds., vol. 3556 of *Lecture Notes in Computer Science*, Springer, pp. 28–37.
- [104] NORMAN, D. *The Design of Everyday Things*. Doubleday, New York, NY, USA, 1988.
- [105] NORMAN, D. Why doing user observations first is wrong. *interactions* 13, 4 (2006), 50–63.
- [106] NORMAN, D. A., AND DRAPER, S. W. *User-Centered System Design: New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, 1986.

- [107] PALMER, S. R., AND FELSING, M. *A Practical Guide to Feature-Driven Development*. Pearson Education, 2001.
- [108] PANDIT, N. R. The creation of theory: A recent application of the grounded theory method. *The Qualitative Report* 2, 4 (December 1996). Available at <http://www.nova.edu/ssss/QR/QR2-4/pandit.html>. Last accessed on 7 Nov 2007.
- [109] PATTON, J. Hitting the target: Adding interaction design to agile software development. In *OOPSLA '02: Proceedings of the OOPSLA 2002 Conference, Seattle, WA, USA, November 4–8* (New York, NY, USA, 2002), ACM Press, pp. 1–7.
- [110] PATTON, J. Improving on agility: Adding usage-centered design to a typical agile software development environment. In *ForUse2003: Proceedings of the Second International Conference on Usage-Centered Design, Portsmouth, NH, USA, October 18–22* (2003).
- [111] POOLE, C., AND HUISMAN, J. W. Using extreme programming in a maintenance environment. *Software, IEEE* 18, 6 (Nov/Dec 2001), 42–50.
- [112] PORTER, S., AND PORTER, J. M. Designing for usability: Input of ergonomics information at an appropriate point, and appropriate form, in the design process. In *Human Factors in Product Design: Current Practice and Future Trends*. Taylor & Francis, London, UK, 1999, pp. 15–25.
- [113] PUNCH, K. F. *Introduction to Social Research. Quantitative and Qualitative Approaches*. Sage Publications Inc., 1998.
- [114] RASMUSSEN, J. Introducing XP into greenfields projects: Lessons learned. *Software, IEEE* 20, 3 (May/Jun 2003), 21–28.

- [115] ROBEY, D., WELKE, R., AND TURK, D. Traditional, iterative, and component-based development: A social analysis of software development paradigms. *Information Technology and Management* 2 (2001), 53–70.
- [116] ROBINSON, H. AND SHARP, H. XP culture: Why the twelve practices both are and are not the most significant thing. In *ADC '03: Proceedings of the 2003 Agile Development Conference, Salt Lake City, UT, USA, June 25–28* (Washington, DC, USA, June 2003), IEEE Computer Society, pp. 12–21.
- [117] RODRIGUES, A., AND BOWERS, J. The role of system dynamics in project management. *International Journal of Project Management* 14, 4 (1996), 213–220.
- [118] ROYCE, W. W. Managing the development of large software systems: Concepts and techniques. In *ICSE '87: Proceedings of the 9th international conference on Software Engineering, Monterey, CA, USA, March 30–April 2* (Los Alamitos, CA, USA, 1987), IEEE Computer Society Press, pp. 328–338.
- [119] RUBINSTEIN, D. Standish group report: There's less development chaos today. *Software Development Times* (March 2007). Available at <http://www.sdtimes.com/article/story-20070301-01.html>. Last accessed on 7 Nov 2007.
- [120] RUDD, J., STERN, K., AND ISENSEE, S. Low vs. high-fidelity prototyping debate. *interactions* 3, 1 (1996), 76–85.
- [121] SCHENSUL, S. L., SCHENSUL, J. J., AND LECOMPTE, M. D. *Essential Ethnographic Methods: Observations, Interviews, and Questionnaires*. Altamira Press, Walnut Creek, CA, USA, 1999.
- [122] SCHWABER, K. Scrum development process. In *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings* (1995),

- D. Patel, C. Casanave, G. Hollowell, and J. Miller, Eds., Springer-Verlag.
- [123] SCHWABER, K. Controlled chaos: Living on the edge. *American Programmer* (April 1996). Available at <http://www.controlchaos.com/download/Living%20on%20the%20Edge.pdf>. Last accessed on 7 Nov 2007.
- [124] SCHWABER, K., AND BEEDLE, M. *Agile Software Development with Scrum*. Prentice Hall, 2002.
- [125] SHARP, H., AND ROBINSON, H. An ethnographic study of XP practice. *Empirical Software Engineering* 9, 4 (December 2004).
- [126] SHARP, H., ROGERS, Y., AND PREECE, J. *Interaction Design: Beyond HumanComputer Interaction*, 2nd ed. John Wiley & Sons, 2007.
- [127] SHNEIDERMAN, B., AND PLAISANT, C. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 4th ed. Addison-Wesley, 2004.
- [128] SHORE, J. Continuous design. *Software, IEEE* 21, 1 (Jan–Feb 2004), 20–22.
- [129] SINGH, S., BARTOLO, K. C., AND SATCHELL, C. Grounded theory and user requirements: A challenge for qualitative research. *Australasian Journal of Information Systems* 12, 2 (2005).
- [130] SMITH, S., AND MOSIER, J. Guidelines for designing user interface software. Tech. Rep. ESD-TR-86-278, The MITRE Corporation, Bedford, MA, USA, August 1986. Available at <http://www.hcibib.org/sam/>. Last accessed on 7 Nov 2007.
- [131] SMITS, H., AND PSHIGODA, G. Implementing scrum in a distributed software development environment. In *Agile '07: Proceed-*

- ings of the AGILE 2007 Conference, Washington, DC, USA, August 13–17 (2007), IEEE Computer Society, pp. 371–375.*
- [132] STRAUSS, A., AND CORBIN, J. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage Publications Inc., 1990.
- [133] SUTHERLAND, J. Agile can scale: Inventing and reinventing scrum in five companies. *Cutter IT Journal* 14, 12 (2001), 5–11.
- [134] SUTHERLAND, J. The roots of scrum: How Japanese manufacturing changed global software development practices. In *Talk presented at JAOO '05: The 9th annual Conference on Software Engineering, Methods and Best Practices, Aarhus, Denmark, September 25–30 (2005)*.
- [135] SUTHERLAND, J., JAKOBSEN, C., AND JOHNSON, K. Scrum and CMMI Level 5: A Magic Potion for Code Warriors. In *Agile '07: Proceedings of the AGILE 2007 Conference, Washington, DC, USA, August 13–17 (2007), IEEE Computer Society, pp. 272–277.*
- [136] SWALLOW, D., BLYTHE, M., AND WRIGHT, P. Grounding experience: Relating theory and method to evaluate the user experience of smartphones. In *EACE '05: Proceedings of the 2005 annual conference on European association of cognitive ergonomics, Chania, Greece, 29 September–1 October (2005), University of Athens, pp. 91–98.*
- [137] SY, D. Adapting usability investigations for agile user-centered design. *Journal of Usability Studies* 2, 3 (May 2007), 112–132.
- [138] TAKEUCHI, H., AND NONAKA, I. The New New Product Development Game. *Harvard Business Review* (Jan/Feb 1986), 137–146.
- [139] TAYLOR, R. N., AND VAN DER HOEK, A. Software design and architecture: The once and future focus of software engineering. In *FOSE*

- '07: *Proceedings of the 2007 Conference on the Future of Software Engineering, Washington, DC, USA, March 20–22* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 226–243.
- [140] THE C3 TEAM. Chrysler goes to “Extremes”. *Distrib. Comput.* (October 1998), 24–28.
- [141] THE STANDISH GROUP. The CHAOS report. Available at URL: www.standishgroup.com/sample_research/index.php, 1995.
- [142] VENNERS, B. Evolutionary design: A conversation with Martin Fowler, part iii. *Artima Developer* (November 2002). Available at <http://www.artima.com/intv/evolution.html>. Last accessed on 7 Nov 2007.
- [143] VERNER, J. M., AND CERPA, N. Prototyping: Does your view of its advantages depend on your job? *Journal of Systems and Software* 36, 1 (January 1997), 3–16.
- [144] VERPLANK, W., FULTON, J., BLACK, A., AND MOGGRIDGE, W. Observation and invention: The use of scenarios in interaction design. In *Tutorial at the ACM Conference on Human Aspects in Computing Systems, Amsterdam, The Netherlands, April 24–29* (1993), ACM Press.
- [145] WHITWORTH, E., AND BIDDLE, R. Motivation and cohesion in agile teams. In *Proceedings of the 8th International Conference on eXtreme Programming and Agile Processes in Software Engineering, Como, Italy, June 18–22* (2007), G. Concas, E. Damiani, M. Scotto, and G. Succi, Eds., vol. 4536 of *Lecture Notes in Computer Science*, Springer, pp. 62–69.
- [146] WHITWORTH, E., AND BIDDLE, R. The social nature of agile teams. In *Agile '07: Proceedings of the AGILE 2007 Conference, Washington, DC, USA, August 13–17* (2007), IEEE Computer Society, pp. 26–36.

- [147] WILLIAMS, L., AND COCKBURN, A. Agile software development: It's about feedback and change. *Computer* 36, 6 (June 2003), 39–43.
- [148] WILLIAMS, L., AND KESSLER, R. All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM* 43, 5 (2000), 108–114.
- [149] WINOGRAD, T. From computing machinery to interaction design. In *Beyond Calculation: The Next Fifty Years of Computing*. Springer-Verlag, 1997, pp. 149–162.
- [150] WITHROW, J., BRINCK, T., AND SPEREDELOZZI, A. Comparative usability evaluation for an e-government portal. *Diamond Bullet Design Report no. U1-00-2. Whitepaper* (December 2000). Ann Arbor, MI, USA.
- [151] WIXON, D., AND JONES, S. Usability for fun and profit: A case study of the design of DEC Rally version 2. In *Proceedings of the Workshop on Human-Computer Interface Design : Success stories, emerging methods, and real-world context, Boulder, CO, USA* (San Francisco, CA, USA, 1995), Morgan Kaufmann Publishers Inc., pp. 3–35.
- [152] WRIGHT, P., BLYTHE, M., AND MCCARTHY, J. User experience and the idea of design in HCI. In *DSV-IS '05: Proceedings of the 12th International Workshop on Design, Specification and Verification of Interactive Systems, Newcastle-upon-Tyne, UK, July 13–15*, vol. 3941 of *Lecture Notes in Computer Science*. Springer, 2006.

