UNIVERSITY OF CALGARY

Agile Methods and User-Centered Design: How These Two Methodologies are Being Integrated in Industry

by

Brian David Fox

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

AUGUST 2010

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled “Agile Methods and User0Centered Design: How These Two Methodologies are Being Integrated in Industry” submitted by Brian David Fox in partial fulfillment of the requirements for the degree of Master Degree in Computer Science.

_____
Supervisor, Dr. Frank Maurer
Department of Computer Science

_____
Co-supervisor, Dr. Jonathan Silllito
Department of Computer Science

_____
Dr. Rob Kremer
Department of Computer Science

_____
Ron Murch
Haskayne School of Business

_____
Date

Abstract

Agile development approaches and User-Centered Design (UCD) both strive to build software that meets the customer's or user's needs. For many software applications, a user interface (UI) that is usable adds value for the users. Recently, there has been some evidence that suggests that practicing agile methods alone does not ensure that an application's UI is usable for the user. As a result there has been interest in combining agile methods with UCD practices. This research presents the results of a qualitative empirical study to contribute to an understanding of how these two methodologies are being effectively combined, The results present a general process model, we call the Agile-UCD General Process Model (AUGPM), and three refinements of the AUGPM. We call these the Specialist Refinement, Generalist Refinement, and the Facilitator Refinement. The results also present when participants felt the UCD practitioner should be brought into an agile methods development process.

Acknowledgements

During the course of my journey through graduate school there were many people responsible for my reaching the end. At this point, I would like to take the opportunity to acknowledge my thanks to these people.

To Dr. Frank Maurer, thank you for all your direction, advice, patience, and above all, your support. Without your kindness and guidance this research would never have been completed.

To Dr. Jonathan Sillito, thank you for co-supervising my work and for all the impromptu, drop-in office sessions that you were always there for. You imparted me with knowledge that I will carry with me for a lifetime.

To all my ASE lab mates Patrick, Robbie, Xin, Yasser, and the German students, thank you all for your assistance and input over the years.

Finally, to Colleen Kim Klemacki, thank you for all your help, support, and careful prodding that helped get me through the difficult times and finish this thesis.

Dedication

This is for you mom. Without all your help and support I would have never finished this journey. Your blind eyes taught me how to see things I otherwise would have never seen.

Publications From This Thesis

Some of the materials and ideas presented in this thesis may have previously appeared in the peer reviewed publication:

David Fox, Jonathan Sillito, Frank Maurer: Agile Methods and User-Centered Design: How These Two Methodologies are Being Successfully Integrated in Industry, Proc. Agile Conference, Toronto Canada 2008, IEEE, p. 63-72.

Table of Contents

List of Tables

List of List of Figures and Illustrations

1.0 Introduction

Most software applications today require some form of a user interface (UI) in order for humans to interact with that application. It makes sense to build a UI that improves the experience of using it in a positive and productive manner for the user. Some companies are realizing that a positive user experience (UX) is essential to the success of their software products. The importance of the usability of a software product has permeated a number of multi-national companies such as Hewlett Packard, IBM and Apple [3]. Hewlett Packard's former CEO, Carly Fiorina, has made the concept of achieving a quality user experience the hallmark of her approach. Karel Vredenburg, IBM's Program Director of Corporate User-Centered Design and User Engineering, has been a long-tem advocate for positive user experience. Apple's innovative and usable design allowed the company to dramatically improve their bottom line. Not only are these companies in favor of providing improved UX for the end user, but many other large corporations are changing and re-thinking their approaches and processes to building better software products using proven heuristic guidelines to develop UIs that provide a better UX for the end user [3].

Developing better software requires some form of methodology to insure that the process of developing that software is not ad hoc in nature. An ad hoc approach may work for smaller projects but simply doesn't work for today's larger projects [1, 2]. Agile methods [1] are an alternative to traditional approaches to building software, aimed at

[1] Agile methods refers to the many similar but not identical approaches that make up this methodology. For the purpose of this thesis "Agile methods" and "Agile methodologies" will be used interchangeably.

satisfying the customer. These approaches strive to deliver better software by involving the customer, or customer representative, closely throughout the development process and by delivering working software in small iterations as quickly as possible. In doing so, the intent is that the customer receives the product that they really want and need. This is achieved by continually being involved in the process and seeing the product first hand [4]. However, recently there has been some evidence that agile software development practices alone do not always ensure that they build software that is usable and what the end user wants and needs [5, 7]. Although agile methods attempt to build better software for the person paying for the software, the customer, this does not mean that it is better software for the person that will actually be using the software, the end user.

Another approach to building software that is more usable is interaction design (ID). This methodology places the end user as a key player in the UI design and development process. The aim of ID is that the software meets the user's wants and needs by employing user-centered design (UCD). Agile methods and ID have the same ultimate goal: to build a product that is what the customer and user wants and needs. However, interaction designers and agile developers approach building software from different perspectives in terms of upfront resource allocation for design. In terms of design, agile approaches address it from the perspective of the code. ID concentrates on design in terms of how the UI is being used by the user to complete the required tasks. Agile methods concentrate on expeditious delivery of working software to the customer with minimal upfront design, whereas ID tends to allocate more time and resources for research and user testing before a single line of code is written.

Because of differences in upfront resource allocation, these two approaches appear to be very different and hostile in terms of cohabitation in the same software project [8]. However there is evidence that these two approaches are being used together in industry today [5, 6, 7, 8, 9] but relatively little is known about how these two software development processes are being integrated in projects.

Our research aims to explore the way teams can combine elements of UCD and agile methods in a way that maintains the benefits of both approaches. Specifically, the aim is to take current research in this area and expand on it to describe a general integration process that is common across multiple teams as well as multiple software companies. In doing so, this research hopes to provide further valuable information that may be helpful to teams in the software industry interested in combining these methodologies in their current process.

1.1 Agile Methods

Agile methods are relatively new software development methodologies that are quickly gaining popularity in the software industry [10, 11, 14]. They are thought of as iterative, lightweight, people-centric processes. The common basis for the numerous approaches of agile software development methodologies came into being in February 2001 with the founding of the Agile Manifesto [1, 12, 13]. The main values of the agile methodologies are:

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

The manifesto also states: “while there is value in the items on the right, we value the items on the left more” [26].

These values are aimed at reducing the heavyweight software development processes that preceded agile methods, such as the traditional Waterfall method, [2] as an answer to “the eager business community asking for a lighter weight along with faster and nimbler software development process” [11, 12]. For the purpose of this thesis, the term heavyweight software development process refers to a software development that relies heavily on processes and tools, comprehensive documentation, contract negotiation, and following a static planning process.

1.2 Interaction Design

Interaction design is aimed at providing a better UX through a blend of analytical and creative abilities that allow the user interface designer to solve problems relating to a UI implementation [15]. The successful implementation of ID means the understanding of the user’s wants and needs by the designer in terms of the product on which they are interacting within the workplace or at home. By emphasizing and understanding the user’s needs, the interaction designer can solve the complex problems that are faced when trying to deliver a product that is a usable one. It is important to note that good software usability is not a single dimension or perspective. Jakob Nielsen [16 pp25] defines a product that is usable as one that should have the following five attributes:

1. Easy to Learn
2. Efficient to Use
3. Easy to Remember
4. Few Errors

5. Subjectively Pleasing

In order to determine if a product is indeed usable, Nielsen suggests that the user, or a representation of the user, must be part of the development process. He states that usability is typically determined by employing a set of test users that are selected to represent the intended users that will perform tasks on the system in question [16]. The above type of testing helps to determine if the computer system meets the user's needs in terms of their interacting with the UI and it is therefore user-centric in nature.

One perspective of computer science aimed at understanding how users interact with computers is the study of human computer interaction (HCI). According to Preece, Rogers and Sharp HCI is:

> " concerned with the design, evaluation, and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them" [17].

HCI includes the practice of user-centered design (UCD), which involves the user throughout the design process. The idea is to build an effective design that is suited to the users needs by including the user in the design equation [18, 20].

Like the waterfall methodology [20], UCD also aims at gathering relatively large amounts of information before construction of the code begins. The idea is to obtain a strong understanding of who the user is and what the user's tasks and needs are. This is achieved in an iterative manner by gathering contextual information, applying it to create low fidelity prototypes and testing those prototypes with the user before any implementation commences. By including UCD practices in the design process, the end product generally has improved usefulness and usability [21].

1.3 Motivation for Research

At a first look, agile methods and ID have opposite approaches in terms of the work that is being done up front before development begins. Agile methods approach software design by eliminating a great deal of the up front resources allocated. For example, one of the most commonly used agile processes is eXtreme programming proposed by Kent Beck and Martin Fowler [22, 23]. The approach used with eXtreme programming is one of short iterations with smaller feature sets and therefore fewer requirements collected before implementation begins.

ID approaches design from the perspective of how the software is used by first spending a considerable amount of time researching the users, their needs, the tasks they need to perform and then iteratively testing a UI design. In his book, "The Inmates Are running the Asylum", Alan Cooper suggests that a fair amount of upfront research be done before any implementation is conceived as an approach to ID [24]. His approach is to address user's behaviors and idiosyncrasies before any development begins.

Although ID and agile approaches are very different, both of these methodologies have the same goal in mind, and that goal is to build better software. As a result of this, there has been an increase in interest directed towards how agile methods and ID can both be employed on the same project to produce a hybrid methodology.

1.4 Research Questions

The purpose of this research study is to determine how, as well as when, Agile methodologies are incorporating ID into the same software development process. In order to explore how and when teams can combine elements of UCD and agile methods in a

way that provides the benefits of both approaches the following research questions are addressed in our research.

- How are these two methodologies being combined given that they have very different up front resource allocation techniques?
- What are the roles of the team members that are directly involved in the process of combining these two methodologies?
- What different strategies are being used to incorporate an ID process and agile methods into the same software project?
- What effect does the integration of these two methodologies have on their original approaches or processes?

By addressing these questions and building on existing research in this area it is the intent of this research to construct a general integration process that can be used across multiple teams as well as multiple software projects. In doing so, this research hopes to provide a generalized roadmap for future software teams to integrate these two methodologies into a single process.

1.5 Road Map

Chapter 2 gives a brief overview of the Waterfall software engineering process. This is used to draw upon the motivation for the advent of the agile methodologies and the similarities with methodologies discussed in this thesis. This is followed by descriptions of agile methods as well as ID, more specifically user-centered design. It then looks at existing reports on combining ID and agile methods.

Chapter 3 describes the research approach used during the study. This includes the processes used to obtain the data and the different techniques applied to the data in order to analyze it.

Chapter 4 presents the research findings in terms of how the two methodologies are being aggregated in industry today. This includes the overall general process that is being used as well as three refinements of that process specific to roles of team members. This section discusses the different strategies that are being employed on software teams and what dynamics emerge as a result.

Finally, Chapter 5 provides a conclusion to the thesis. It includes an overall conclusion, a discussion of the findings, limitations of the study, as well as future work that may be of value.

2.0 Background

This chapter provides background information for these software methodologies that are directly associated with this research topic. Specifically these are a common traditional software approach and the two methodologies that are the focus of this research, agile methods, and Interaction Design (ID).

First, we discuss a traditional approach to software development, the Waterfall methodology. This is intended to give the reader an idea how software has been delivered in the past as well as the possible motivation for the focus of this research: agile methods.

This chapter then discusses the agile approach to software engineering as an alternative to the traditional Waterfall approach. We specifically look at two of the more popular processed of the agile space, Scrum and XP. Following the agile methods section is an overview of ID and, specifically, User Centered Design (UCD). Next, we investigate some of the issues with the amalgamation of Agile methods and ID. Finally, a review of existing work is presented. This review will point out previous research contributions that this thesis will build on as well as the perspectives not covered in that previous research that are covered in this research study.

2.1 The Waterfall Approach

In an attempt to improve on ad hoc approaches to software development, the notion of disciplined software engineering methodologies arose [1, 28]. The aim was to make software development process more predictable and efficient and, hence, adding some order of control over the process. Fowler refers to this as planned design [1]. A

disciplined model provides the software developers with a roadmap to produce the required software in a highly structured manner.

One traditional engineering methodology for software development is often called the "Waterfall" methodology first proposed by Royce in 1970 [2]. This methodology suggests following a predefined path, or steps, through seven sequential stages of software development, gathering system requirements, gathering software requirements, analysis, program design, coding, testing, and operations.

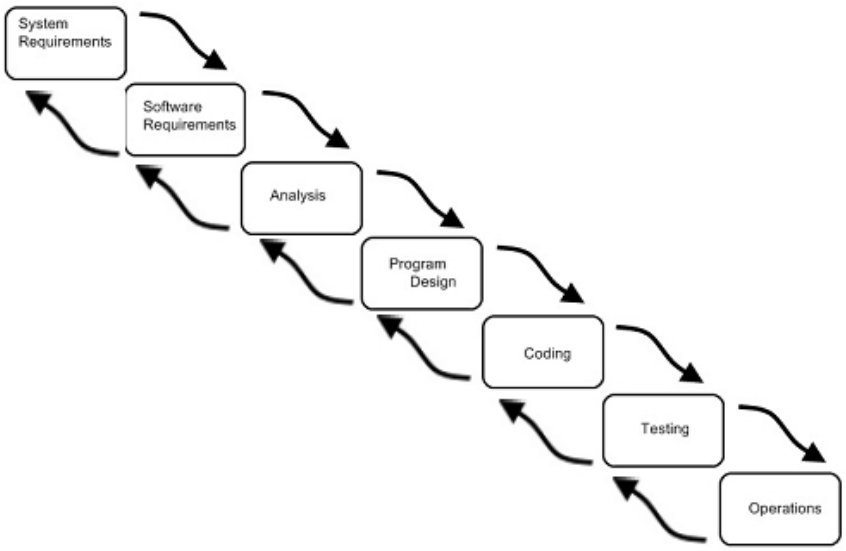

Figure 1: The seven steps from the Royce Waterfall methodology

One property of this methodology is gathering all system and software requirements, completing application analysis, and completing software design before any coding begins [2]. This leads to the allocation of large amounts of upfront documentation, planning, and design prior to development.

However, Waterfall methodologies "have not been noticeable for being terribly successful. They are even less noted for being popular" in terms of an approach by the

individuals building software using this methodology [1]. This is possibly because of their intensive bureaucratic structure that slows the development process down and adds an extensive amount of documentation, which results in a considerable amount of up front resource allocation.

The Standish Group's Chaos Report of 1995 suggests that in the United States 31.1% of traditional process software projects will be cancelled before completion. The Standish group also estimated that in 1995 on average only 16.2% of software projects were completed on time and on budget [25]. This suggests that alternatives to traditional software development processes are needed in order to improve software development project success. Historically there were several models available between Royce's waterfall model and the emergence of agile methods including the Spiral model amongst others [75]. We use the waterfall model in this thesis as the extreme opposite to agile methods to highlight the differences between the two methodologies.

2.2 Agile Methods

In February 2001 in the Wasatch Mountains in Utah, 17 people from various software development backgrounds gathered to describe an alternative to document driven, upfront resource heavy software development processes such as the Waterfall approach [12]. What emerged was the Manifesto for Agile Software Development from several individual methods including Scrum and XP amongst others. This workshop defined the common ground for all these approaches and gave them a common joint name which was agile methods. The aim was to uncover "better ways of developing software by doing it and helping others do it" [26].

This new software development approach was geared toward being more adaptive and receptive to change [1] rather than following the detailed rigid plan conceived prior to development [2]. In contrast to working in a Waterfall like linear manner, agile methods favor an iterative process characterized by a succession of incremental small releases containing a smaller set of features for that specific iteration [6]. This approach allows for changing software requirements discovered during the iterative process. Close customer contact during the iterative development process supports this flexibility.

Iterations string together mini-projects to form the bigger application over the course of the time of the software project. At the beginning of the two to four week iterations, the requirements for the products features are gathered by the developers from the customer representative. The requirements are prioritized and selected for that iteration. At the end of the iteration the features are demonstrated for approval [8].

By building these mini-projects, agile methods seek to produce finished code that is more valuable for the customer than a document. Larger projects could be split into these mini-projects that are started, finished and tested, and delivered to the customer on a regular basis [27].

Because developing software is different from project to project, the context of the development situation changes from project to project in terms of processes and project environment. The principles behind the Agile Manifesto aims to help software development to succeed in changing and unpredictable environments that are common-place today [14]. The Agile Manifesto was conceived by industry people from varying software processes aiming at combining these different processes into agile methods. Some of these processes include Crystal, RUP, eXtreme Programming, Lean

Development, Scrum, and Feature Driven Development [26, 30]. Two of the more popular agile methods components being practiced today are eXtreme Programming and Scrum [8, 14] and were used by the participants in this study. For this reason the next two sections will briefly look at these two processes.

2.2.1 eXtreme Programming

The first of the two specific practices used by participants in this study used was eXtreme Programming. Kent Beck first coined the term, eXtreme Programming (XP) in his 1990 publication, "eXtreme Programming Explained: Embrace Change". Beck explains, "XP is a light-weight methodology for small-to-medium-sized teams developing software in the face of vague or rapidly changing requirements" [22]. XP can be further broken down into components, values and practices.

XP originally had four values [22] and later in his second edition Beck introduced a fifth value [31]. These values are open communication, simplicity in design, constant feedback, courage to address change, and respect for all members on the team. Beck states:

> "We want to do everything we must do to have to have stable, predictable software development".

To do this Beck uses the five values above to build a discipline of software development through a set of twelve activities or practices from the first edition of Extreme programming explained [22] and adds an additional thirteen practices from the second edition [31]. These practices are employed in order for software teams to have

some control and guidance over the work they do [22]. The practices that were most relevant in the course of this research were:

- **The Planning Game** This is combining technical estimates with the overall business priorities in order to quickly determine the scope of the next release.
- **Small Releases** Small releases are used containing the requirements with the highest business value for the customer.
- **Simple Design** The correct design for software is a design that will:
    - Will run all of the tests
    - Does not contain duplicate logic
    - Asserts all intentions essential to the programmers
    - Has the minimal amount of methods and classes necessary

  "Every piece of design in the system must be able to justify its existence on these terms"
- **Testing** All program features must have, as well as pass, automated tests to be considered part of the application.
- **On-site Customer** Having a customer collocated with the development team aids the developers in terms of answering questions relating to the product design, resolving differences that arise, or dealing with feature prioritization.
- **Whole Team** A team should include people with all the skills necessary for the project to succeed.
- **Incremental Design** Designing regularly to make improvements to the design helps reduce the cost of changing the system.

To build better software, XP also has certain roles that team members play. People on an XP team should fit the role they have and the rest of the team should be aware of that individual's role at any given time [22]. The idea is that everyone contributes the best he or she has to offer to the team. The key roles Beck mentions are the programmers, customer, and the tracker/coach. These were evident in the roles that were being employed by teams in this research study.

2.2.2 Scrum

The second practice the agile teams participating in this study employed was Scrum. The Merriam-Webster defines Scrum as [32]:

1. **a:** a rugby play in which the forwards of each side come together in a tight formation and struggle to gain possession of the ball using their feet when it is tossed in among the; *also* : the arrangement of players in a scrum **b:** usually a brief and disorderly struggle or fight

Ken Schwaber, a co-founder of the Scrum approach to software development, defines Scrum as "an iterative, incremental process for developing software in chaotic environments" [37].

Scrum is a process that was first used in Japan for hyper-product development in 1987 by Ikujiro Nonaka and Hirotaka Takeuchi [33]. The process was called Scrum because of the close resemblance to some of the qualities of the game of rugby. Both the game and the development process posses the ability to be adaptive, quick, self-organizing, and have few rests [34].

Scrum is a team management and control process that strives to cut through the complexity and allows focus on building software that meets business needs [35]. The

process focuses primarily on the team level, allowing them to control the building of software in the manner that they choose [34]. The roles on a Scrum team are the Scrum master which is a management role, the product owner, and the Scrum team, which is typically made up of cross functional members that bring the necessary skill sets to a project. [36]:

Scrum teams work in an iterative manner during the software development process. These iterations are called Sprints. A Sprint is a fixed period of time in which the Scrum Teams work on completing the commitments of the Sprint goal. The Scrum Team is free to accomplish this goal as it sees fit. Typically, a Sprint is thirty calendar days. During the Sprint, the team has two mandatory accountabilities; daily Scrum meetings must be attended by all Scrum Team members and the Sprint backlog must be kept up-to-date and reflect an accurate and evolving picture of the team's work.

The diagram below represents the overall scrum process from inception of the product vision, the creation of the Product Backlog, the Sprint Planning Meeting, the selection of the Sprint backlog, the Sprint and finally the Sprint Review meeting.

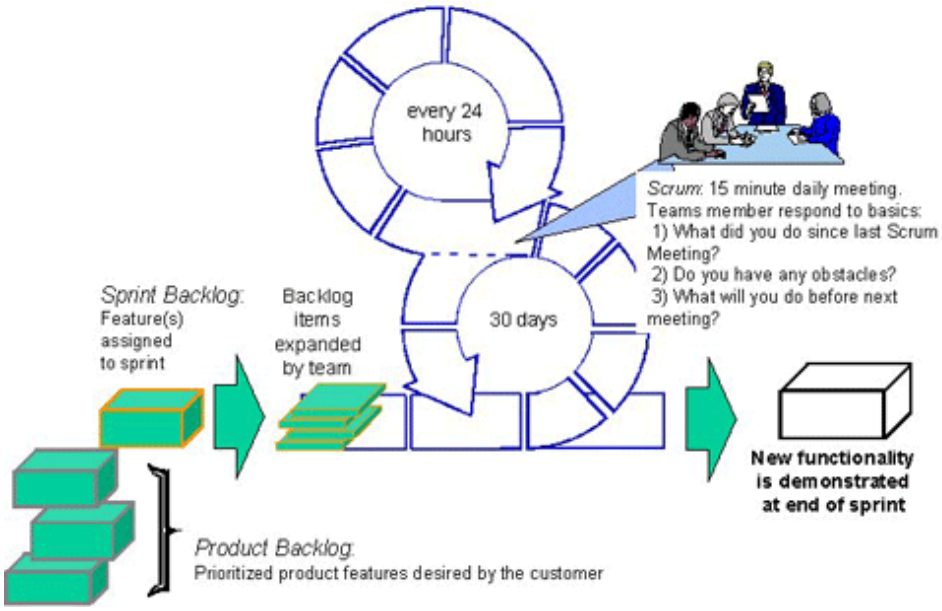

Figure 2: Overview of the Scrum process [74]

In the next section, this thesis discusses Interaction Design.

2.3 Interaction Design

According to Cooper "Interaction Design is a tool for knowing what the user wants" and knowing what the user wants allows for building software that is of value to the user as well as the company building it. He also states that in doing so you will reach your market and enjoy better consumer loyalty [55]. Rogers, Preece, and Sharp define Interaction Design as "designing products to support people in their everyday and working lives" [17]. Bacon states that Interaction Design is a specialized design field that practices iterative problem solving using a blend of analytical and creative abilities to aid people in achieving what they want or need to do during work or play [39]. However, just adding functionality to an application in an attempt to help the user complete their daily tasks does not mean it is more useful. In his book, The Design of Everyday Things,

Norman remarks "The same technology that simplifies life by providing more functions in each device also complicates life. This is the paradox of technology [40]. The overall user experience is also important, or as Nielsen states, using a product should be subjectively pleasing [16].

Defining what user experience is has proven to be difficult with many different definitions arising from different sources [41]. For example, Alben's definition of user experience is: "All the aspects of how people use an interactive product: the way it feels in their hands, how well they understand how it works, how they feel about it while they're using it, how well it serves their purposes, and how well it fits into the entire context in which they are using it" [42]. Hassenzahl and Tractinski describe the user experience as a consequence of a user's internal state (predispositions, expectations, needs, motivation, mood etc), the characteristics of the designed system (e.g. complexity, purpose, usability, functionality, etc) and the context (or the environment) within which the interaction occurs (e.g. organizational/social setting, meaningfulness of the activity, voluntaries of use etc) [43].

To better understand user experience issues there has been much study for some time by the HCI community [44]. One approach to HCI is User-Centered Design (UCD). The participants in the current study all integrated UCD into their development process and the following sections discuss the HCI and the UCD approach to UI design.

2.3.1 HCI

The Curricula for Human-Computer Interaction defines HCI as: *"human-computer interaction is a discipline concerned with the design, evaluation, and implementation of*

interactive computing systems for human use and the study of major phenomena surrounding them" [45]. HCI strives to enhance the interaction between humans and computers and make technology easier for them to use [18]. This implies that HCI strives to make software applications usable.

2.3.2 Usability

As discussed in section 1.2, Nielsen's definition of usability had five attributes. He states [16, pp-25]:

> "*Only by defining the abstract concept of usability in terms of these more precise and measurable components can we arrive at an engineering discipline where usability is not just argued about but is systematically approached, improved, and evaluated (possibly measured)*".

There is an increasing awareness in the software industry that usability is a quality attribute and should be addressed during development [46, 47, 48]. There are a number of HCI techniques available that help developers deal with usability issues that appear during the software development process [46]. This thesis looks specifically at one such HCI practice, UCD, which was employed by the participants in this study to mitigate usability issues in their software development process.

2.3.3 User-Centered Design

Vredenburg et al. present their working HCI definition of UCD as [49, pp 471-478, 50, pp-12]:

> "UCD is herein considered, in a broad sense, the practice of the following principles, the active involvement of users for a clear understanding of user and

task requirements, iterative design and evaluation, and a multi-discipline approach".

Barnum contends that UCD is [20, pp 187-188]

> "*a shift in product design away from a validation of features and capabilities to a focus on the user's perceptions of usefulness and feelings of satisfaction has come a change in terminology from product usability to user-centered design*".

While Constantine states: [54, User-Centered Approaches]

> "*UCD in practice is a rather cluttered collection of loosely related techniques and approaches having in common little more that a shared focus on users, user input, and user involvement. While it may be different things in the hands of different practitioners, at its core, user-centered design is distinguished by a few common practices: user studies, user feedback, and user testing*".

UCD is based on three kinds of design activities [20, pp 187-188]. The first involves an early focus on users and tasks, in order to understand the users, the tasks they perform, and the environment in which the tasks are performed. The second set of activities involves empirical measurement of product usage to provide information about how easy is it to use, how easy is it to learn, and any other usability issues relating to the use of that product. The final activity involves,

> "*iterative design that fixes the problems found by the users in usability testing as part of the product development life cycle*".

In his early work, Gould contends there are four principles for the usability design process in terms of HCI [18]. These are:

1. Early and continual focus on the user.

2. Early and continual user testing.
3. Iterative Design.
4. Integrated Design. Every aspect of usability should evolve in parallel.

It is clear the above principles are being translated into the software industry. This was very clear in the processes carried out by participants in this study as well as these same principles being implemented in other large companies in industry. In 1999 IBM alone had "*25 UCD labs worldwide with a total of 78 lab cells*" [52, pp 67-71]. Mao, Vredenburg, Smith, and Carey conducted a survey with 103 respondents with UCD related backgrounds that were currently employing UCD practices. Their survey findings suggest that using UCD methods generally improved the end product. They also found that Contextual Inquiry, Iterative Design, and Usability Evaluation were considered the most important practices [50, 51].

The above principles and practices, suggest two main practices used in UCD, contextual inquiry and usability testing, both of which were employed by participants in this study.

Contextual inquiry refers to a designer taking an ethnographic study approach to better understand the users of a particular product [17]. Raven and Flanders, among others, suggest that Contextual Inquiry is used to understand your users or audience [18, 20, 53]. Typical activities include contextual interviews, observation, reconstruction of previous events or tasks performed, and discussions with the users.

Usability testing, as the name suggests, refers to testing a software product with users or user representatives to determine how usable a software product is. This process can be broken down into two parts, prototyping and testing.

According to Nielsen, it is essential not to design a full-scale implementation of a product based on early designs [16]. Instead Nielsen advocates using quick, cheap, throw-away prototypes of the system which can be used for early evaluation of the system. These prototypes can be developed quickly and hence can change many times over the course of design until a better understanding of how the UI will work is achieved. There are five central aspects to prototyping [16, 17]:

1. Vertical Prototyping refers to the amount of working functionality that a prototype has. The more functional features a prototype has, the greater degree of Vertical Prototyping it has.
2. Horizontal Prototyping is tied to reducing the functionality while increasing the features. A prototype with a large number of features with little or no functionality has a high degree of horizontal prototyping. Figure 4 demonstrates both Vertical and Horizontal Prototyping.
3. Low Fidelity Prototyping refers to working with low tech mediums such as paper and pencil, sticky notes, story cards, cut-and-paste drawings or any medium that is quickly produced and therefore can be easily discarded. Low fidelity prototypes are an example of a high degree of horizontal prototyping and a low degree of vertical prototyping.
4. Medium Fidelity occurs when a prototype has both features and functionality of the finished product. At this level, typically the UI has been partially developed features and some features that have not been developed or are non-working. This would be a good example of the prototype having some Vertical and Horizontal Prototyping properties.

5. High Fidelity Prototyping refers to a prototype that is very close to, or possibly is, the finished version of the working UI. It has most or all of the functionality of the intended application. This would be a good example of the prototype having a high degree of both Vertical and Horizontal Prototyping properties.

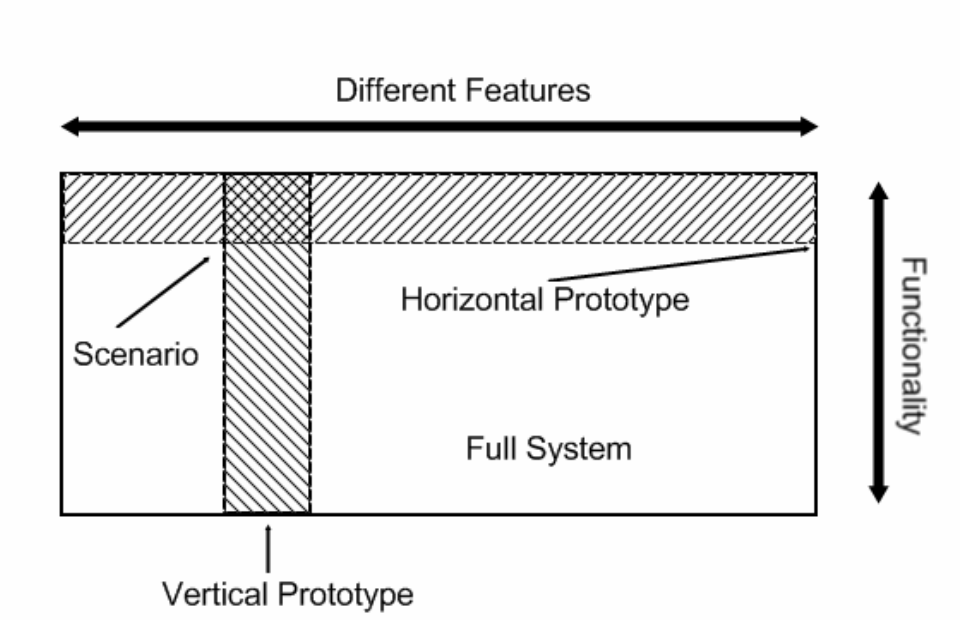

Figure 3: Vertical prototyping representing functionality versus horizontal prototyping representing different features. The example above is shown where the two meet and consists of performing some task and the functionality to perform that task represented by the cross hash in the small box[72, 16]

Usability testing can take on many forms. Traditionally in the late 1980's UI testing was expensive and time consuming [20]. Typically there were between 30 and 50 subjects, under the observation of psychologists, used to ensure that data was statistically valid. Given the time and the professionals involved in this type of study, cost often made it prohibitive for smaller scale projects. To deal with this, Nielsen came up with the notion of "discount usability". The idea behind discount usability is to focus on what your resources will allow. Using some usability testing practices is better than none at all [16].

Perhaps the simplest, inexpensive, and most popular method of gleaning feedback is heuristic evaluation. Heuristic evaluation is one of Nielsen's methods of discount usability [20].

> "*Heuristic evaluation is a method for finding usability problems in a user interface design by having a small set of evaluators examine and judge its compliance with recognized usability principles (the 'heuristics')"* [56].

To perform this type of evaluation, an evaluator critiques an application against a list of predetermined heuristic guidelines. This greatly reduces errors in design before any formal testing techniques are employed. This technique is performed without end users present. Once a heuristic evaluation has been completed, formal testing can commence [20].

One form of formal testing is cognitive walkthroughs.

> "*A cognitive walkthrough is a usability inspection method designed to evaluate ease of learning, particularly by exploration*" [20, 57].

Typically a user is asked to complete a predefined task in order to determine if the outcome is what the evaluator expects [16].

Another commonly used approach to usability testing is called ***Wizard of Oz*** testing. This is little more than a test subject interacting with a mock user interface which is being manipulated by "*the man behind the curtain*". For example, when a user is interacting with paper prototypes the testing facilitator switches between pages to simulate UI functionality. This greatly reduces cost in terms of making a fully functional hi-fidelity prototype [16].

Focus groups, another valuable technique, typically consist of between six and twelve people chosen from a group of people representing typical users [17]. The group can spontaneously discuss or react to an application's design before or after it is built. The information gathered can be used by the developer to hash out a better design. One of the dangers in using this particular technique is that there is no guarantee that the users know what they want or if what they want is even possible [24]. However, as Nielsen proclaims, "*some data is better than no data*" [16, pp 23-27] and the above danger is therefore worth the risk.

During the course of this research all of the participants practiced some or all of the above techniques during their software development process.

2.4 Agile Methods and User-Centered Design

Recently there has been some interest in how UCD and Agile methods can be used together on the same software development project [5, 6].

The Extreme Programming vs. Interaction Design debate between Kent Beck, the father of eXtreme programming, and Alan Cooper, a prime proponent of interaction design, added interest in this area. Cooper argues that interaction design should be completed before the actual code is written. Beck, on the other hand, argues that completing all the interaction design up front creates a bottleneck for the developers and that development should begin before design is complete [59]. However, section 2. discusses acquiring a small set of requirements for a small set of features that can be implemented as quickly as possible. From a textbook perspective both of these

methodologies seem to be complete opposites. However, these two processes do have some similarities.

For example, both processes are human centric, as discussed in sections 2.2 and 2.3. Agile methods strive to build customer satisfaction by keeping the customer in the development loop. This way the customer is constantly seeing the features and functionality as they are rolled out for approval. Interaction design, on the other hand, attempts to provide improved user experience by building a more usable user interface. Another similarity is that both these methodologies are iterative in nature. However, they both iterate over different artifacts. Agile methods iterate over application code, where interaction design iterates over user interface design.

In order to understand how the above concerns have been addressed, it is important to examine existing related work. The following section addresses this.

2.5 Related Studies

Recent research examines the integration of UCD with Agile methods to determine if and how these two approaches can cohabitate on the same software project. The publications discussed here show what previous work has been done in the area that is pertinent and directly related to this research study. They are briefly compared with some of our findings that are later discussed in more detail in the Findings section of this thesis, page 54.

Patton discusses the motivation for adapting UCD practices into his agile development process [7]. His development team was using eXtreme Programming (XP) practices similar to the practices being employed by participants in this study. The team

included expert end-users that were product managers working for the same company. They were building software using an agile approach at an aggressive rate with "*deliveries that were on time and the expected scope intact*." However, they found that the resulting product had features that the end-user did not want and was missing features that the end-user did want. In an attempt to correct this problem, Patton describes a ten-step ID process that he and his team added to their current development process to produce an Agile Usage Centered Design process.

His process identified roles that were included in the design process [7]. However his process did not include a UCD specialist (UCDS). Instead, his team was responsible for the UI design and the UCD practices associated with that process. This approach was similar to one of the approaches discovered in our research. We found that some of the teams our participants worked with did not have a UCDS on their team before and during development. Instead their teams were responsible for the UCD practices which seemed very similar to Patton's approach.

Overall, Patton stated that he and his team were satisfied with the results of their process. He does not claim that his process builds better software but rather that it did leave his team with valuable tacit knowledge. What his publication does not cover is the actual process being implemented throughout the entire lifecycle of that software project from beginning to end. Instead his publication provides a snapshot into a portion of his software lifecycle process where ID is implemented and not in the project's entirety. This is an area that this research intends to build on and provide some insight into.

Patton's process also closely resembles Constantine's ten-step usage-centered design process that was originally conceived as a lightweight Agile UCD approach [60].

Constantine suggests that his process is lightweight in terms of typical UCD upfront resource allocation. Constantine's process may be lightweight as opposed to previous UCD processes, but his process still champions doing all the design work upfront. His paper outlines the ten-step process but lacks empirical evidence or case studies directly tied to his process [60].

In their practitioner's report, Meszaros and Aston [5] describe the process they used in including usability testing into an Agile methods project. Like Patton's approach, Meszaros and Aston did not include a UCDS role on their team.

Meszaros and Aston had developers on their team acting as the UCDS. Their approach included typical UCD practices such as low fidelity prototyping and usability testing to determine a final UI design. This process meant that the developers were also the UCDS like some of the participant's teams in our study [5].

Meszaros stated that adding UCD practices was of value to his development process. In fact he makes the claim that is partially the motivation for this research:

> "*Emergent design doesn't work well for user interfaces when using Agile practices alone. Some design up front seems to provide better guidance to the development team and provides earlier opportunity for feedback*" [5, pp 289-295].

His paper discusses the implementation of their process over a short period of time - which does not support an overview of their complete software development process lifecycle.

Ferreira, Nobel, and Biddle investigate four different projects that employed Agile methods iterative development and UI design [9]. Their approach is qualitative in nature using grounded theory similar to the approach used in the current study.

Their paper briefly describes the roles of the team members in the four projects [9]. In projects one, two, and four, the teams consisted of developers and a UCDS, case three had a developer acting as the UCDS. In the three cases that employed a UCDS, the UCDS interacted with the customers to derive the interface design requirements. In the case where no UCDS was used, one of the developers, whose main interest was UI design, interacted with the customer.

The paper very briefly discusses the processes and the roles that the different teams used. However, given the length of that particular publication, providing detail was not possible. More detail may have given more insight into a generalized approach of all four projects integrating these two processes [9].

In her publication "Interaction Design and Agile Development: A Real-World Perspective", Ferreira [58] goes into much greater detail of the Agile Interaction design integration. This study dealt with 9 teams integrating Agile methods and ID. She discusses two main design strategies, the Comprehensive and the Evolutionary strategy.

The Comprehensive strategy refers to the big design up front (BDUF) wherein a very large part of the UI design is completed before implementation begins. In fact she states that

> "*Teams that subscribed to this view completed their UI design such that it was a representation of the entire system under development"*.

However, in our study, we found that none of our participants performed extensive upfront UI design for the entire system prior to development.

The Evolutionary strategy refers to teams coding and designing small sections of the application at a time. It then slowly evolves into the finished design and application. She states that

> "*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*".

This strategy closely resembled the findings that the participants followed in our study.

Ferreira also discusses 3 implementation strategies, the Refinement, the Parallelization, and the Looking Ahead strategy.

The Refinement implementation strategy was typically used in conjunction with the Comprehensive design strategy. Most of the UI design work was done upfront and the minor details or changes in the design were completed during the iterations that followed. In her words

> "*the comprehensive UI design created up front was refined during the iterations, successfully transforming it into an 'implementable' interaction design"*.

Although our study did not show that the participants were employing Ferreira's Comprehensive design strategy, all of our participants did employ a refinement factor into their design implementation process.

The Parallelization implementation strategy referred to the development implementation and UI design being done in parallel.

"*As the developers iteratively implemented the system, the UI was iteratively designed and evaluated separately*".

In other words interaction design is done in parallel with development. This was confirmed in our research. Ferreira also states that

"*applying the Parallelization strategy appeared to be independent of whether or not a comprehensive interaction design had been created up front"*.

Although none of the participants in our study were employing comprehensive interaction design up front, all of them did somewhat follow the Parallelization implementation strategy.

The Looking Ahead implementation strategy refers to the ID person(s) designing iteration(s) ahead of the development implementation schedule. Ferreira states

"*Looking Ahead implementation strategy was characterized by the interaction designers creating a design for a fixed number of iterations ahead (usually one or two iterations ahead) of the developers implementing the current iteration*".

This is another facet of Ferreira's work that closely matched the findings with our participants. All of the UCD specialists or acting UCD specialists designed at least 1 iteration ahead of the development iteration. In some cases it was 2 iterations ahead. The Looking Ahead implementation strategy findings also match those of Sy [6] that we discuss below.

Although Ferreira's work outlines a great deal of detail in terms of the strategies used it does not abstract those findings to an overall general process based on the similarities. Our thesis looks at abstracting the similar facets uncovered in the data to generalize a higher-level model of the agile methods development and UCD integrations.

Ferreira's thesis also concentrated heavily on the processes and strategies whereas our study concentrates on individual team members roles and how they affect the processes and strategies of agile methods and UCD integration.

Finally, Sy describes the process of integrating UCD with agile methods currently being successfully adopted by Autodesk [6]. The findings in this thesis closely resemble Sy's process described in her publication. In Autodesk's process, Sy refers to iterations on design as "*cycles*". Cycle zero is used to acquire initial information about the project by conducting a contextual inquiry. According to Sy, contextual inquiry refers to a designer taking an ethnographic study approach to better understand the users of a particular product. It is important to note that our study addresses the interaction between UCD and development practices and not the business decisions that are made prior to design. For example, we do not take into consideration business feasibility studies as part of the actual design process [6]. Typical activities include contextual interviews, observation, reconstruction of previous events or tasks performed, and discussions with the users [17].

If the team is refining an existing product, cycle zero is used for "*the alignment of all team members' understanding*" and for developing an overall vision for that project. If it is an ongoing project, the UI design is derived by performing UCD testing on the previously completed implementation cycle along with the previously performed contextual inquiry. An initial design is conceived and sent for implementation by the developers in cycle one [6].

Once in cycle one, the UCDS designs prototypes and conducts usability testing to refine the design for cycle two as well as conducting contextual inquiry for cycle three.

Upon the completion of cycle one, the implemented code is passed from developers to the UCDS and the UI design is passed to development for implementation for cycle two [6, pp 112-132].

"*This pattern of designing at least one cycle ahead of the developers, and gathering requirements at least two cycles ahead, continues until the product is released*"

allowing development and UCDS to work in parallel throughout the projects cycles [6].

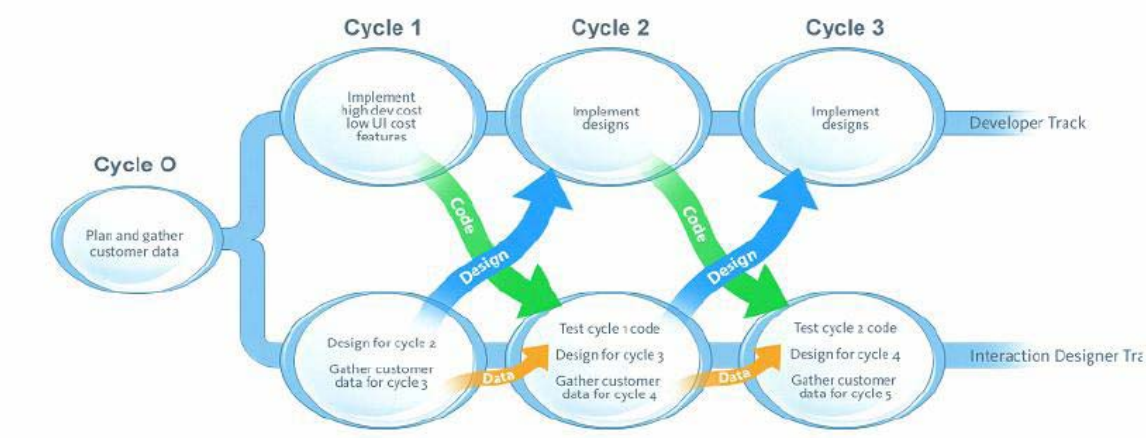

Figure 4: An illustration of Sy's Autodesk Agile UCD integration process [6].

Sy's paper provides a detailed view of the process that is used at Autodesk. The process is very much in keeping with some of our findings in terms of a general approach to integrating these two methodologies. We discuss this further in our Findings section.

However, although the description of the process is detailed, it is the study of how one company is adapting these two methodologies and is therefore restricted to that company. Does this mean that the approach she is using is adaptable for other companies? In our study we investigate methodology integration based on multiple companies to determine if it is successful in more than one company.

All of the above previously related works are similar in that they all target how agile methods are integrating usability practices into their software development processes. There has been other previous work that involves bringing UCD practices into a software development project. This is described in section 2.5.1 in more detail.

2.5.1 Adding UCD Practitioners

Another aspect uncovered in this research was that adding UCD and the UCD practitioners to a project is not without its challenges and strategies. The following related studies serve as a comparison to some of our findings. Because at this time there is little or no published research in terms of when to bring a UCD process and practitioner into an agile project, we use the following studies only to briefly draw comparison with our findings which are discussed in more detail in the Findings section.

Venturi and Troost investigate how the User-Centered Design approach is being integrated into industry settings. The research data consisted of web surveys taken by 83 UCD practitioners with UCD experience from 3 to 15 years employed at companies that varied in size. Sizes spanned from large corporations (Financial, Computer, Telecommunications etc.) to small consultancies. Their research was aimed at determining critical issues that enable usability practitioners to avoid usability outcomes that may be poor. One issue was management understanding the value of UCD practices in a project.

According to Venturi and Troost, their findings also showed *"UCD is particularly employed in big companies."* However, their findings were not specific in terms when the UCD persons were brought in to the project. However, the authors also claimed that there is a very small ratio of UCDS versus the overall number of employees but this topic is

left for later work [69]. Although this work shows that the majority of project managers believed that UCD should be part of the design development process, it does not specifically discuss when the management feels it should be included in a project. Our findings addressed this question. The work also does not specify if the individuals surveyed were on agile teams or more traditional teams. Our study only dealt with agile teams.

Gulliksen et al. surveyed 192 usability professionals in Sweden to determine some of the development processes that involved UCD [67]. The software processes used on the projects were predominantly RUP, a variant of RUP, or XP. Their study showed that there were 5 key factors regarding what is needed for usability practices to be effective on a project. They were that usability be part of the project from the start, support form project management, support from the users, support from management, and acceptance from the developers. This was consistent with what we found with some of the participants in our study as well.

Gulliksen et al. also see education and raised awareness of UCDS inclusion as an important facet in the success of usability practices being integrated into software development. This was also a common factor that we discovered I our research.

Although this work uncovers interesting findings, it does not address strategies for usability integration in any detail. In other words. although most of the participants believed that usability should be part of the process they did not specify if that it was used all the time nor when usability team members were brought in to a project [67].

Bruno and Dick's work elicited data from 14 usability practitioners in a qualitative study using a grounded theory approach [71, pp 261-264]. Their aim was to try to

understand what was critical in providing good usability in a product in an industrial setting according to the individuals responsible for employing the usability practices. The major finding from this research was

> "*that an iterative usability process of research, design and evaluation stages needs to be implemented though the project lifecycle*".

Their findings also showed the importance of "*Evangelizing*" usability practices to other team members to gain credibility. The outcome was a finding that speaks to including an iterative usability process including research, design and evaluation stages throughout the lifecycle of a product. These three practices were also used by the teams throughout their development process uncovered in our research.

However, they did not compare other usability strategies in terms of issues relating to adding these practices after the project had begun. In other words, the paper does not discuss any of the problems related to bringing a UCDS into the project after it had begun [71]. Our study, on the other hand briefly explores some of the problems UCD practitioner participants encountered when they were brought onto an agile project after that project had begun.

Finally, Rosenbaum presents a paper that describes the organizational approaches and usability practices that HCI professionals consider to be of value in terms of increasing usability research within companies [68, pp 337-344]. The researchers gathered data through surveys from 134 HCI professionals at 3 conferences in order to gain an understanding of the insights and advice related to specific obstacles offered by participants. Rosenbaum et al. discuss the notion of "*strategic usability*". They define this as

"*Embedding usability engineering in the organizational processes, culture, and product roadmaps. In strategic usability, usability data contributes to corporate-wide decision-making, such as product priorities and make vs. buy decisions*".

Rosenbaum's data revealed that the obstacles preventing greater strategic impact of usability engineering within the organizations was based on lack of communication and the education of its value.

Although the paper speaks to "*strategic usability*" the strategies in the paper are not specific to the timeline for when usability is added to an agile process. Their findings do, however, speak to some of the issues pointed out by some of the UCD practitioner participants in our study. For example, lack of early involvement, lack of knowledge and understanding, and resistance to UCD practices being brought onto a project, were all concerns or issues that were revealed by the participants in our study as causes for some resistance in terms of UCD agile practices integration.

2.6 Chapter Summary

This chapter presented the background necessary to understand our study. It investigated the traditional approach to software development and its successor, agile methods, to demonstrate why this newer approach to software development is being used in ever increasing instances. The chapter broke down agile methods into its two most popular facets being practiced in industry today; eXtreme Programming and Scrum. This was followed by a high level discussion of Human Computer Interaction which was followed by a brief discussion of usability and User-Centered Design. Next, the chapter

discussed the possibility of agile methodologies cohabiting with User-Centered Design on the same software development project. Finally, this chapter discussed the existing related work in terms of how a UCDS was being added into existing software development projects to determine if the same issues arise on agile projects. Chapter 3.0 discusses the research method used in this study and presents the collected data.

3.0 Research Method & Data Findings

This chapter discusses the research approach taken by this study. We discuss why qualitative research is suited well for this study. Next a brief overview of the grounded theory approach is given. Following this, the data collection process is described. The chapter continues with a description of how a grounded theory approach was applied in this study. This includes the three coding processes applied to the data; open, axial, and selective coding. Finally, the validity of the study is discussed followed by a conclusion.

3.1 Qualitative Research

In the past, there has not been a universal definition of what qualitative research is [61]. For the purpose of this thesis, the definition put forth by Gorman and Clayton will be used and it states [62, pp-1]:

> "*Qualitative research is the process of inquiry that draws from the context in which events occur, in an attempt to describe these occurrences, as a means of determining the process in which events are embedded and the perspectives of those participating in the events, using induction to derive possible explanations based on observed phenomena."*

The main strength of qualitative research is its ability to be subjective with the essence "*to capture life as it is being lived*". Qualitative researchers argue that the human experience cannot "*be described using numbers or can adequately be explained by manipulating, measuring, or controlling variables*" [63, 566-569].

The research approach reported in this thesis was qualitative in nature. It was clear from the beginning that a quantitative approach would not be possible. Quantitative

research requires controlled environment experiments and relies on data gathered from a random sampling of participants. This study relied on data derived from processes that occurred in non-controlled environments, such as industrial software teams. The other consideration that made qualitative research a fit for this study was that, because participants with Agile and UCD experience were difficult to find, they were not randomly selected.

Because every software project is different [10] and to get "*a full understanding of a social system like a software development, one needs qualitative research in order to get a holistic understanding of what the important factors are and how they may influence*" [64, pp 1-6]. For these reasons, the choice to use qualitative research was the most appropriate for this study.

3.2 Grounded Theory

As mentioned in Section 3.1, the decision was made to use a qualitative approach for this study. More specifically, we choose a grounded theory approach. The grounded theory approach consists of iterative data collection and analysis with the goal of producing a theory to explain a situation of interest. Powel defines Grounded theory as:

> "*Studies that seek to inductively and systematically develop taxonomies and theories through intensive analysis and coding of descriptive data about phenomenon under investigation; theories emerge through iterative, constant comparison of concepts and categories against data said to be grounded in given naturalistic settings being investigated*" [61, pp 91-119].

Because very little information existed in terms of how agile methods and UCD were being integrated, I had little idea of how these processes were being integrated and, thus, had no preconceived idea of how this was being done going into the study. Grounded theory relies on the researcher beginning a study without preconceived theories; instead it lets the researcher analyze the data to construct the theories or find themes [66]. This is achieved in an iterative manner by gathering data, applying coding procedures to the data, and analyzing the data. These concepts are discussed in the following sections.

3.2.1 Data Collection: The Interviews

In order to ground concepts in data, first the data must be gathered. One way to achieve this is with participant interviews that can be transformed into meaningful data by way of transcription and coding procedures. Coding procedures allow the researcher to [66]:

- Build rather than test theory
- Provide researchers with analytic tools for handling masses of raw data
- Help analysts to consider alternative meanings of phenomena
- Be systematic and creative simultaneously
- Identify, develop, and relate the concepts that are the building blocks of theory

In order to acquire the data, the study conducted in-depth, semi-structured interviews either face-to-face, by telephone or using a computer application, Skype. The interviews included 13 participants from Canada, the United States, and Europe. The interviewees were recruited through networking with professionals in academia, industry, and the Yahoo Agile-Usability users group.

A semi-structured interview combines predefined, structured, open-ended questions with the flexibility to ask subsequent questions during that interview. The interviews were conducted at the interviewee's convenience. Interviews performed lasted between 28 and 62 minutes. The predefined questions were meant to be very broad in nature in order to provoke further more detailed questions. The predefined questions progressively got finer in terms of granularity. The questions divided the interview into four main portions. The four main questions were:

- What is your background in terms of employment, or project types you have worked on, and education/training?
- What are the activities surrounding your software development approach?
- How are the activities put together to fit into your software development process from beginning to end?
- How did you include agile methods/ UCD in your process?

Each of these four questions always lead to a subset of questions conceived in terms of the answer given by the participant that varied from participant to participant. New derived questions were added to interviews that followed as the interview/data analysis iterations continued. These questions arose when the participant made an interesting remark during the interview and were then added to the sub list of questions and used in the interviews that followed.

For example, when P5 was asked what are the activities surrounding your software development approach a portion of her answer was:

> ” *There is sort of an ‘us and them’ type of mentality that seems to exist. You do get Agilest*[2] *who have worked successfully with usability people or design people. But frequently you get that sort of we know what we are doing and you get these usability people coming in and they slow us down*”.

The interesting part of this data was the comment referring to ‘us and them’. This led to the addition of the question; is there tension between the UCDS and other team members? This question proved valuable in that it provided some social context between team members. P5 was one of the very early interviewees and as a result that question was used in all of the following interviews.

An audio recording of each interview was made and each recording was transcribed verbatim to be used in the coding procedures. In order to speed the transcription process up considerably, Express Scribe was used. Express scribe is a transcription tool that allows the user to hotkey various features such as rewind, fast forward, play, or stop etc.

[2] For the purpose of this thesis the term Agilest refers to a practitioner of Agile methods.

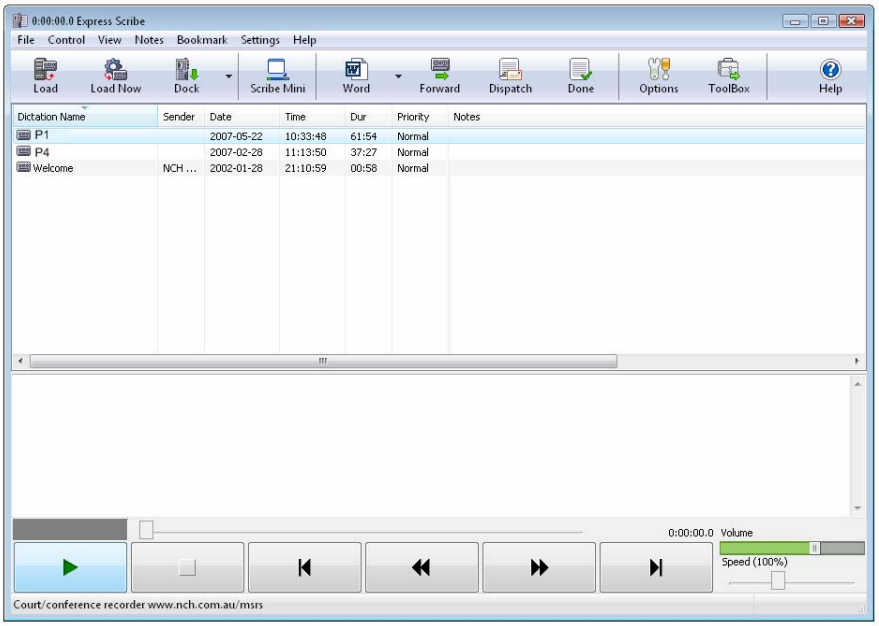

Figure 5: A screenshot of Express Scribe transcription tool

Once the interviews had been conducted with the participants and transcribed, the next step was to code the transcripts. The coding procedure had three phases, open coding, axial coding, and selective coding. These coding procedures are further discussed in sections 3.2.3, 3.2.4, and 3.2.5.

3.2.2 Participants

The following section discusses the participants interviewed for this study. For the purpose of this thesis the participants will be referred to as P1 (participant 1) … P15 (participant 15) and will be referred to as PN in an inline sentence and [PN] as a reference.

Originally there were fifteen participants interviewed for this study. Each of the participants was from a different team and company with the exception of two of the participants. P3 and P6 worked for the same company but on different teams and in

different countries. P12 and P14 were removed from the study for different reasons. P14's interview .wav file was corrupted and the data lost and P12 did not meet the criteria of working in an agile methods team that was required for this study.

The 13 included participants, from which the data was analyzed, had varied backgrounds and various roles on their teams. Participants were from Canada, the United States and Europe. The companies they worked for varied in size from smaller software consulting firms to very large multinational computer software firms.

P1 was a UI designer on with ~ 8 years of experience practicing UCD with no formal development training. P1 worked with a medium sized multinational company.

P2 was a developer and UI designer with ~15 years of experience. P2 worked with a large multinational company.

P3 was a developer with ~ 13 years of experience. P3 did not have formal UCD training, however, claimed to be a self-taught usability practitioner. P3 worked for a medium sized multinational company.

P4 was a developer with a formal HCI background acting as an information architect with ~ 15 years of experience. P4 worked for a large multinational company.

P5 was a Human Factors Engineer and a UI designer practicing UCD with ~7 years experience with no development training. P5 worked for a medium size multinational company.

P6 was a business analyst with ~5 years of experience. P6 did not have formal UCD training, however, started his/her career in a usability lab for a large multinational company. P6 worked for a medium sized multinational company. As mentioned above P6 worked for the same company as P3 but in different country.

P7 was a Human Factors Engineer with a development background, practicing UCD with ~11 years of experience. P7 worked for a large multinational company.

P8 was a developer with ~21 years experience. P8 did not have formal UCD training but claimed to be self-taught. P8 worked for a small consulting company located in a single country.

P9 was a Human Factors and Engineer with ~8 years of experience. P9 worked for a large multinational company.

P10 was a developer with formal UCD training with ~4 years experience. He worked for a medium multinational company.

P11 was a developer, UI engineer with UCD practices, and an information architect with ~19 years of experience. P11 worked for a large multinational company.

P12 was a developer/ architect and scrum master with ~11 years of experience. P12 was working for 2 different consulting companies.

P13 was a UCD and information architect. It is important to note that only a portion of P13's data was transcribed due to portions being inaudible for transcription. For the purpose of this thesis the term medium sized multinational company refers to a company with more than 100 employees and less than 1000 with locations in more than one country, a large multinational company refers to a company with more than 1000 employees with locations in more than one country.

Table 1 below summarizes participant's role, experience level, and company size.

Table 1: Participant's role, experience and company size.

Participant	Role(s)	Experience	~Company Size
P1	UI Designer/UCDS	~8 years	medium
P2	Developer/UCDS	~15 years	large
P3	Developer/UI Designer	~13 years	medium

P4	Developer/UCDS/Information Architect	~15 years	large
P5	UI designer/UCDS	~7 years	medium
P6	Business Analyst/UCDS	~5 years	medium
P7	Human Factors Engineer/UCDS	~11 years	large
P8	Developer/UI Designer	~21 years	small
P9	Human Factors Engineer/UCDS	~8 years	large
P10	Developer/UI Designer/UCDS	~4 years	medium
P11	Developer/UI Engineer/UCDS	~19 years	large
P12	Developer/Architect	~11 years	n/a
P13	UCD/Information Architect	~7 years	large

3.2.3 Open Coding

Our analysis involved various coding activities. We first performed open coding. Straus and Corbin define this activity as "*the analytic process through which concepts are identified and their properties and dimensions are discovered in data*" [66]. This coding is "fluid and dynamic" and consists of attaching specific code words, developed during the open coding process, to discrete incidents in portions of the data. This is done on the first pass through the transcription that represents raw data. The goal of open coding is to identify ideas and or concepts in the data and attach a code. It is here that the mass of data begins the first segment of the data organization process.

In order to facilitate open coding on the data, HyperRESEARCH was used. HyperRESEARCH is a "*code and retrieve data analysis program*". It allows the researcher to tag codes directly onto the transcription for later analysis. In the case that new codes were generated by a later transcription, they were entered into the HyperResearch application and appeared in the center panel. The application's center panel, shown in figure 7, displayed the codes from all the transcriptions and therefore could be applied on a first or subsequent passes through of the data. Adding new codes to previously coded transcriptions typically happened in the next phase of analysis, Axial

coding. If a new code was added to the code list in the HyperResearch application and that code also fit the existing coded tex, it was added to that text by the researcher.

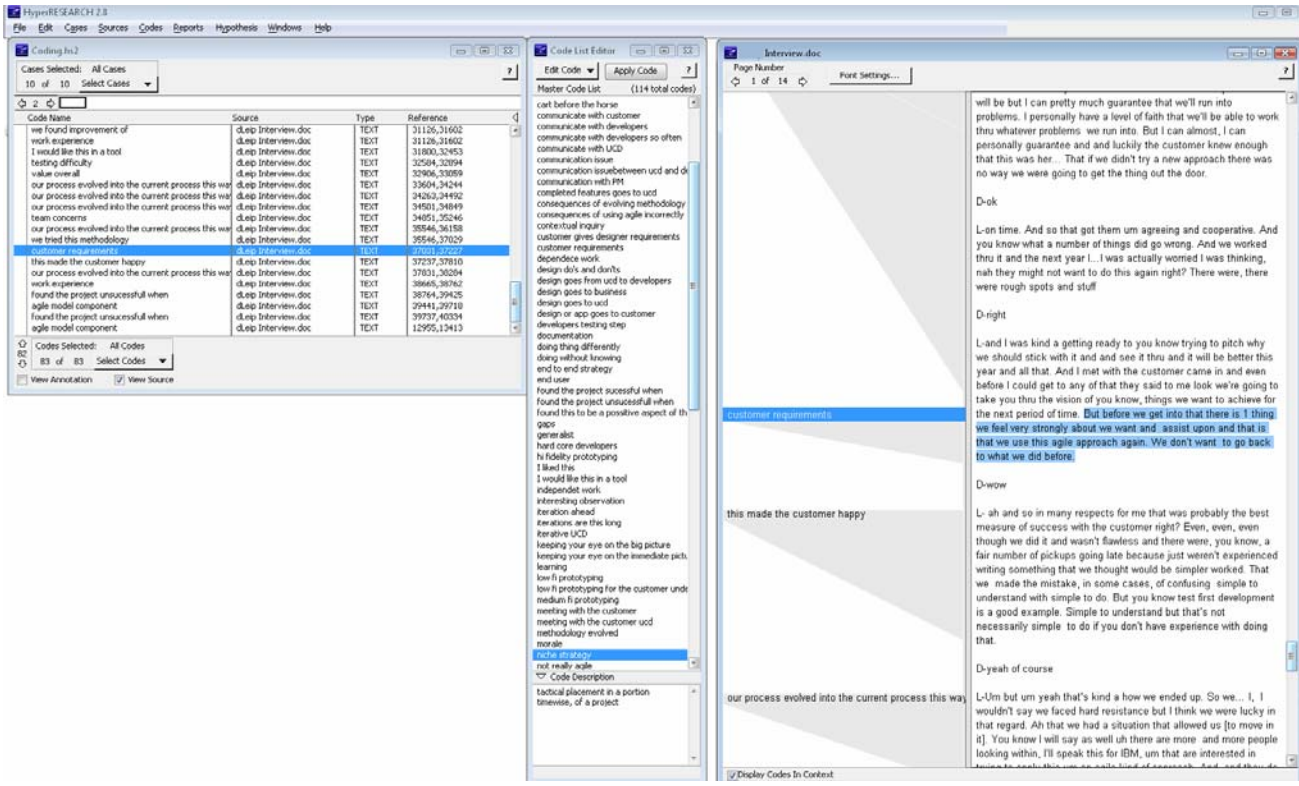

Figure 6: A view of HyperRESEARCH coding screen. The blue areas (from left to right) represent the order in which codes are applied to a transcription, the overall code list, and finally where the code is applied to which specific text in the transaction[3]

In total, 126 codes were created during the open coding process. These codes are listed in Appendix B.

3.2.4 Axial Coding

The next stage of analysis is axial coding. Trauth defines axial coding as [65, pp 115]:

[3] HyperRESEARCH can be found at http://www.researchware.com/hr/index.html

"*The stage where categories and relationships between categories are supposed to emerge. It is also the stage that the open codes are grouped into categories and subcategories, and indeed some open codes become categories in their own right*."

Axial coding assembles the previously attached codes from the open coding stage into core relationships to each other [66]. This is achieved through constant analysis and comparison in terms of the participant's interviews. These relationships, or categories, then act as a guide to precipitate adding newer codes to the previously coded transcripts if applicable. For example while making a second pass through, a newer code may be added to that transcript if it is applicable.

After iterating through the attached open codes and comparing their similarity and frequency of their use, the following categories emerged:

- Upfront predevelopment stage. – resource allocation(s)
- Roles – who is doing what
- Compromises – who gives up what
- Passing Design Around – when the design (UCD and development) changes hands
- Tools Used
- Team Dynamics
- Customer/Developer/UCDS/Team/Communication
- Testing
- Existing and Evolving Methodology/Practices

- Requirements Process
- Project/Methodology Results
- Miscellaneous

For example, the Roles category was derived from the codes that pointed back to discussion in the transcripts relating to roles in the development process. These codes from one category are shown in Table 2.

Table 2: Open codes that have been assigned to the Roles category during Axial coding

CATEGORY	CODES
Role – who is doing what	Customer voicing requirements Dependence work – dependencies or non-dependencies in a team environment End user – their role/ description Customer role Role as a generalist Gorilla tactic – this is the user's explanation of their practice process Hard core developers – roles or a culture in the team? Working independenly of the team Roles We are this as a team Who works on what –roles Work experience

The full table of Open codes that are applied to these categories can be found in Appendix C.

3.2.5 Selective Coding

The final coding process in the grounded theory approach is selective coding. Corbin and Strauss define selective coding as "*the process of integrating and refining the theory*" [66, pp 247].

In this stage, the core categories are used to derive a small set of the high-level concepts that form the big picture, questions, and or themes that emerge from the data [66].

From twelve initial categories, two higher-level categories emerged. These categories emerged as a result of further abstracting the original 12 categories into more general categories that showed common characteristics in order to derive an overall theme. These categories were Overall New Combined Agile/UCD Process, and Roles. The 12 initial categories were assigned to one or both of the new categories with the exception of the miscellaneous category. The miscellaneous category contained some codes that could not be easily categorized into the new categories or they simply did not fit those categories. For example one code from the miscellaneous category was "Working on projects that weren't really agile". This code did not fit into either of the newer categories or offer any beneficial data to the study.

From these two new categories came the main theme derived from our grounded theory approach: team roles were a driving force of the participant's new processes. In other words, each of the participant's methodologies for agile UCD integration had two

parts, the process and the roles on the team driving that process. This was true in every participant's approaches to Agile UCD integration.

Table 3: The two high level categories and assignment of the existing categories to them.

NEW CATEGORY	INITIAL CATEGORIES
Overall New Combined Agile/UCD Process	• Upfront predevelopment stage. – resource allocation(s) • Compromises – who gives up what • Tools Used • Customer/Developer/UCDS/Team/ Communication • Passing Design Around – when the design (UCD and development) changes hands • Testing • Existing and Evolving Methodology/Practices • Requirements Process • Project/Methodology Results
Roles	• Customer/Developer/UCDS/Team/ Communication • Passing Design Around – when the design (UCD and development) • changes hands

	• Team Dynamics • Roles – who is doing what

This prompted the development of the Agile/UCD General Process model (AGPM). This model discusses the processes used to integrate these two methodologies discussed in detail in Section 4.0.

3.3 Study Validity

In order to ensure the validity of the study, a number of steps were taken. One issue with this type of research is to ensure that the data was correctly gathered. To ensure this, two steps were taken. First, all the data/interviews were recorded as .wav files for later playback and transcription. This ensured nothing was missed during the interview. Second, the transcriptions and recordings were carefully compared at the time of transcription. As stated above, a majority of the interviews were performed over the phone or using Skype. There were instances where small portions of the interviews were inaudible due to technical difficulties such as static on the line, background noise, or low volume levels. In these cases, this section(s) of the interview was given a '[na]' tag and that data was not used.

Another issue to be considered is the sample size. Although the sample size was small the participants all worked on different teams. Six of the participants worked for medium-to-large multinational companies that had multiple teams around the world using their development and UCD integration techniques [P2, P3, P4, P5, P11, and P12]. It is

also important to note that the sample size was not relevant to the qualitative approach as we were not striving for statistical validity.

3.4 Chapter Summary

In this chapter, the research method used to collect and analyze the data was discussed. First, qualitative research was discussed from a high level perspective. Next, why qualitative research was a fit for this study was also discussed. The chapter then took a more detailed look at the research approach used, grounded theory. This included the three coding techniques; open, axial, and selective coding. Following this, an overview of the interview structure was given in order to understand the initial data gathering process. Next, the participants in the study were described in terms of their roles with their respective companies, their experience levels, and the size of the companies they worked for. Finally, the validity of the research approach was discussed.

Throughout the chapter, some of the raw data was introduced without a detailed discussion about that data. A detailed discussion of that data is presented in Section 4.0.

4.0 Findings: Compromising Methodologies

During the course of this research, a number of findings were discovered in terms of how agile and user centered design methodologies actually accomplish coexistence in a software development environment. I developed a general integration process model for integrating these two methodologies with three different refinements to that model.

This chapter presents these findings in the following manner: First, the Agile-UCD General Process model is presented followed by the three refinements of that model. Next, we discuss the compromises that needed to be implemented by both methodologies. Finally, some of the challenges are presented of when user-centered design is brought into an agile development process.

4.1 The Agile-UCD General Process Model

The participants had similar approaches when integrating agile methods and UCD. We call this approach the Agile-UCD General Process Model (AUGPM). The AUGPM is a representation of the participants approach to integrating agile methods and UCD from a very high level perspective. In this model, we show the common practices that all the participants followed. The AUGPM consists of two main process sections, the Initial Stage, and the Iterative stage. It is in these two stages UI design is conceived and implemented by people in different roles on their teams. The AUGPM has two main aspects that were derived as a result of the grounded theory approach presented in 3.2. These aspects were the process and the team roles driving that process. We then discuss

the more specific process refinements used by the different team roles used to solve this problem, all of which were based on the AUGPM.

The study showed that the participants processes all had an upfront stage for UI design that occurred before any software development began on the UI. For the purpose of this thesis, we will call this the "Initial Stage". This is not to say that during the Initial Stage no other development on the project had begun at all. For example, P8 stated that while the UI was being designed, the developers were working on *"some of the technical things, the back end, the database or some of the technical infrastructure"* and not *"really focusing on flushing out the high value* [UI] *features"*. P9 claimed that projects that she had been brought into from the beginning did have some of the backend design and implementation done previous to her arrival. One participant stated that development had started on the design before the UCDS had started their design but it was an isolated case and not the norm in terms of their development process.

> "*Unfortunately there were other times when the developers started the development but we didn't know that the design work had to be done yet. So it was really miscommunication higher up* [referring to the business owners] [P4].

In the Initial Stage, which typically lasted 2 weeks, we found two main activities, contextual inquiry and low fidelity prototyping/testing. The contextual inquiry preceded the low fidelity/testing activity. One participant described these two activities as the "*discovery stage followed by a prototyping stage*" [P1]. These two activities take place in Iteration 0 in Fig 7. In each of the participant's processes, the initial stage had "*some measure of user research"* [P3]. A User-Centered Design Specialist (UCDS) or a team member acting as a UCDS carried this out. The UCDS performed contextual inquiry to

better understand who the users were and what were their needs in terms of the tasks they needed to perform.

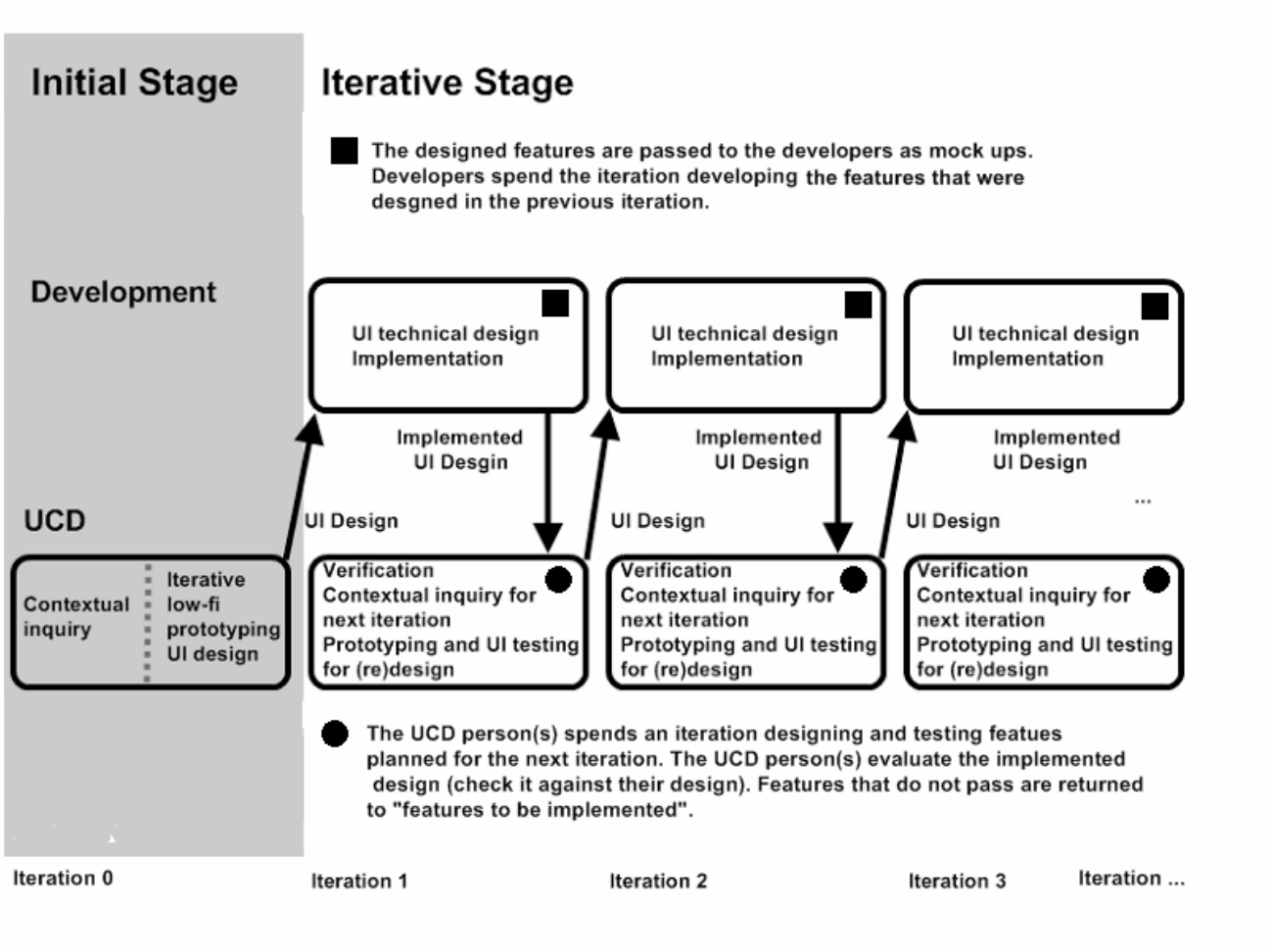

Figure 7: A UCD Agile methods project development life cycle common to the participants. The grey area (Initial stage) represents the upfront UI design stage that happens once in the development lifecycle of a project. The area in white represents the Iterative stage which continues for the rest of the development lifecycle.

The contextual inquiry was followed by low fidelity prototyping activities that varied slightly depending on the team. This low fidelity prototyping consisted of everything from producing hand drawn "*sticky notes on the whiteboard*" [P6] to "*putting together some wire frames to help flush out the requirements*" [P4].

The low fidelity prototypes were constructed in an iterative manner. During each of these iterations, usability testing was performed using real users or team members

acting as users. P3 remarked that their process for testing without real users, in order to discover flaws in the design, was:

> *"For us this was looking at some general UI guidelines and following them. Kind of a collaborative inspection process."* [P3]

The purpose of the usability testing was to identify and correct any usability issues before the initial UI design was handed off to the development team.

> *"And sometimes it was a very light weight way for us to do that* [testing before handing the design off]. *We also tried some paper prototypes as well."* [P12]

Some of the participants initially relied strictly on low fidelity prototyping in their design/testing process

> *"We will do path centered design. When we start HCI, we are focusing very much on path analysis… we will do a very high level prototype, paper and pencil, white boarding, no more than that."* [P7]

As a result, this shortened the upfront UCD design process and, hence, a smaller set of features was derived for the first development iteration. One participant remarked that their process had an Initial Stage that lasted two weeks [P8], while another stated theirs had lasted only a few days [P3].

> *"So the idea that I found out is the quickest way to get up and running is, is to create a set of personas start up with a well defined strategy phase* [Initial Stage]

which you know can be typically anything from a couple of days to a couple of weeks" [P11].

However, the data analysis showed that for most participants the Initial Stage lasted approximately two weeks. The output from the Initial Stage was complete and initial UI design was complete which consisted of low fidelity prototypes and a list of features or requirements.

After the initial UI design is completed, it is passed to the development team initiating the second stage of development, the Iterative Stage. This portion of the development process is illustrated as Iteration 1 through Iteration … in Fig 4.1. Once the development team has the initial UI design, a planning meeting is held to determine which of the features will be implemented in the first iteration. The remaining features are moved to the backlog for future iterations.

The members that attended these planning meetings varied from team to team. On the teams that were part of very large international companies, the planning meetings had larger attendance. Attendees included the developers, UCDS, graphic designers, information architects, the customer representatives as well as different levels of executives and stakeholders

"We evolved it [the planning meeting process] *a bit with certain groups that were always involved in planning meetings. They were the developers, the business analysts and quality assurance"* [UCD members] [P12].

On the teams that were part of smaller organizations, often only a part of the core team and the project manager or customer representative attended the planning meeting. Participant P9 remarked that, on one of her projects the planning meetings were attended only by the project manager, a senior developer, and herself.

Once it is established which features would go into the initial iteration, the development team produces a technical design and development begins. While the development team implements the features for Iteration 1, the UCDS continues with more contextual inquiry, prototyping and UI testing to be used for the next iteration. In other words the development team and the UCDS team work concurrently. The UCDS team basically prepares for the planning meeting for the next iteration. Once the UI features have been completed, the implemented UI design is passed back to the UCDS for verification and usability testing.

Verification consisted of determining if the development team had followed the design rules set out by the UCDS [P1, P5]. Participant P5 claimed that the reason for the verification step was *"just to make sure the grid is respected"* by the development team [P1]. In other words, the rules and guidelines of the UI design put in place by the UCDS were followed and not violated by the development team. P5 said that the verification step was necessary because the "*developers don't have a UI designer with them at all times to keep them honest*" in terms of the design intended by the UCDS.

Usability testing, which typically followed verification, may or may not include user participation. One participant remarked that when users were not available for testing, all the team members did a collaborative UI inspection in order to determine if the user's tasks were possible to accomplish in an effective manner [P3]. Nine

participants (P2, P3, P5, P6, P8, P9, P10, P11 and P12) said they used actual end-users to verify and test implemented features.

If the implemented features are correct and pass the usability tests, they are marked as finished features and await release to the customer. The time period before finished features were released to the customer varied from two days [P9] to three or four weeks [P3].

On the other hand, if a feature failed verification or usability-testing, the UCDS redesigned the UI features and passed them back to the development team for re-implementation in the same iteration. If an issue was uncovered that was too large to be fixed in that iteration *"then it would have to go into another iteration"* [P9]. Once the iteration is finished, the UI design that was developed concurrently by the UCDS is passed to the development team and the process begins all over again (as shown in Iterations 2 and 3 in Fig 7. This process continues iteratively for the duration of the projects lifecycle.

4.1.1 Tandem Development

The above commonalities that are present in the integration of UCD and agile methods could not exist without the developers and UCDS personnel working together in tandem, as illustrated in Figure 7.

As mentioned in Section 4.1, the UCD initial set of features was smaller than that in a traditional UCD process. This meant that design was done incrementally in terms of the multiple smaller sets of features being designed over multiple iterations by the UCDS

and hence being implemented in different iterations by the development team. In doing so, this compromise generated a tandem development approach. Both teams were working on the same iterations but at different times.

In other words, the UCD group member(s) would be designing the UI one or two iterations before the developers started implementation on those iterations. P5 stated they were always working ahead of the development team to produce a design for the developers to follow. P9 stated that

> *"Basically I would work with the product* [the UI design] *2 iterations ahead of development"* [P9].

The UCD group member would also be testing the design after the development team had implemented the design for correctness as well as to ensure no usability issues existed. If an issue was found, it was passed back to development.

> "*They would clean up whatever testing found within the iteration"* [P9].

This means that regular interaction is required between these two member groups throughout all the iterations. Although they were both working on the same product, it was on different portions of the product at different times. We call this Tandem development.

Figure 7 closely resembles Figure 4, Sy's process diagram. Both show the initial upfront stage of UI design as well as the iterative parallel UCD design and developer working on a tandem timeline. This means that our work reaffirms previously existing work published by Sy and shows that Sy's approach is not limited to only her company.

The general approach describes similarities of the approach taken by development teams and the UCD specialists in this study. These similarities represent the interaction between the two during the integration of UCD into agile methods. However, as mentioned above, we also found differences in participant's approaches. Three slightly different approaches for integrating UCD into Agile methods emerged. The differences lie in who executes certain aspects of the process, i.e. who performs which roles in the process. The following sections discuss the approaches used by different teams and the roles of group members that facilitated those approaches.

4.2 AUGPM REFINEMENT 1: THE SPECIALIST

We call this first refinement the *Specialist*. It consists of three main member groups[4]: the users/customers[5], the UCDS, and the development team. Four of the participants, P1, P5, P9 and P10, practiced a form of the Specialist approach. These four participants were on teams with a single UCDS and multiple development team members.

In the Initial Stage of the Specialist approach, the contextual inquiry and the low fidelity prototyping steps are both conducted by the UCDS "*interfacing with the*

[4] For the purpose of this thesis the term "main member group" will be defined as the individual team member(s) that work directly on the software application on a regular basis. An example of a main member(s) would be the developers building the software application. An example of a non member in a group would be a stakeholder that might only see the product rarely. P4 remarked that the high level stakeholders, or upper level executives, only saw the product once a year. They are not considered to be main group members.

[5] "Customers" refers to anyone that has a vested interest in the development of the project as a stakeholder or someone representing a stakeholder. "Users" refers to people that are actual end-users of an application.

customer" [P5] with an almost total absence of the development team. During this time the UCDS is *"*learning the basic requirements for the customer*"* [P1] through contextual inquiry. Once the UCDS has gathered contextual information from the user, the initial low fidelity prototyping work begins. This step involves producing low-fidelity drawings and/or wire frames of a UI's features and testing those with users to determine features for implementation. Low fidelity prototyping tools ranged from pen and paper, sticky notes, and white boards to applications like PowerPoint and Visio. Because the above activities do not typically involve the developers, the UCDS initially acts as a *"bridge role between the developers and the customer"* [P5]. They relay what the user's requests and needs are to the development team.

> *So currently I am in the situation where I am going to the customer's site. I sit in on their meetings. I understand how they work and I translate some technical requirements and some flow of how the screens should be* [P1].

After iteratively prototyping the UI design, an initial high-level UI design is created and the UCDS meets with the development team to ensure that the design is technically possible. If the design is not technically possible, the necessary changes are then made to the design by the UCDS under the direction of the development team. If the initial design is technically possible, it is passed to the development team for implementation and the initial stage is complete. P1, P5, and P10 stated that the Initial Stage typically lasts two to four weeks depending on the project but the timeline was typically two weeks. Whereas, P9 stated this typically lasted six weeks. P1, P9, and P10

stated that once the Iterative Stage begins, the UCD iterations were shortened to two weeks as opposed to up to six weeks in the Initial Stage.

While the development team implements the features for the iteration, the UCDS conducts more contextual inquiry with the user or customer for the next iteration. P9 remarked that "*sometimes I look ahead one or even two iterations to conduct contextual inquiries*". This is consistent with Sy's approach used at Autodesk where the UCDS on her team also looks 2 iterations ahead. Concurrently, usability testing occurs during this time to augment, extend and refine the feature list for the next development iteration. P10 remarked that, in this way, the UCDS continues to work in parallel with the development team.

Once the development team completes their technical design and implementation, they pass it back to the UCDS. This varied in terms of when it was exactly passed. It depended on the team and the project they were working on. The UCDS then takes that implementation back to the customer or user to perform usability tests. This testing differs from usability testing of low fidelity prototypes in that those tests were testing future features, whereas the current tests evaluate completely implemented features. If the implemented features are free of usability issues and they meet the user's approval, they are marked as complete and the iterative stage starts again with a planning meeting to determine the next set of features.

P7 was also a UCDS. However, he did follow the above approach with one difference. P7 typically worked on iterative projects that were riddled with *"red tape"* from the stakeholders. This required him to gather requirements in a longer iterative

process typically 6 weeks long. He did however follow the Specialist refinement in that he was the bridge between the developers, users, and customer.

4.3 Refinement 2: The Generalist

Next we discuss the *Generalist* refinement to the AUGPM. The Generalist refinement uses only two main member groups or roles, the users/customers and the developers acting also as UCD specialists. It is worth noting that the developers were not formally trained UCDS. They were self-taught.

> *"I wasn't working in the lab, but I did get to know the head of the usability lab in Toronto quite well. And he invited me in to watch some user design sessions in action. Which is quite interesting and it peaked my interest. So I am not formally trained at all in HCI but through reading and a big interest in it have gotten to be very involved in that field"* [P6].

The developers acting as UCDSs did have some informal or self-taught UCD expertise. P3 and P8 both stated that their experience with UCD had come from a weekend seminar pertaining to UCD or related reading they had done on their own. This means that some of the developers [P8] or all developers [P3, P6] were responsible for development as well as some or all of the UI design and used a UCD approach. Unlike the Specialist approach, this approach had more than one team member acting as the UCDS present on the team. The least number of team members acting as UCDS we found on a team was two [P8]. The other teams had multiple acting UCDS with most of the team members having some input into the UCD design and testing process. P3

remarked that this was their "*collaborative inspection process*". The UCD activities did vary depending on the team. P8 practiced low fidelity prototyping and prototype testing as well as usability testing after implementation. However, P8's contextual inquiry was limited due to the short initial stage timeline of two weeks. P3 and P6 practiced contextual inquiry, low fidelity prototyping and testing as well as usability testing. Three of the participants, P3, P6, and P8 followed a form of this process.

When developers acted as UCDS, the Initial Stage lasted two to four weeks and included contextual inquiry, prototyping and user testing [P3, P6, and P8]. On average, it was shorter than in the Specialist approach.

The number of developers acting as UCDS varied from team to team. In the case of P8, not all of his team participated in UI design and UCD activities. In this case, P8 was responsible for passing the UI design to the developers that were not participating in the UCD process. In case of P3's and P6's team, all developers contributed to the UCD activities. They may not have initially designed the UI but they were responsible for other activities such as testing the UI. Once the developer has the contextual information that is needed for the first iteration, low fidelity prototyping is started in an iterative fashion either with the customer [P8] or with team members acting as customers to determine the stories for the development iteration [P3, P6]. If customers were not available for usability testing, each member of the team was expected to participate in testing and heuristic evaluation of the UI portion they were building [P6]. This was very similar to the verification performed in the Specialist approach.

Once the initial low fidelity prototypes are tested the UI design is finished. A planning meeting is used to prioritize which features are going in to the next iteration and

the work is split among the developers. This initial design is passed to the developers to implement the features and the Iteration Stage begins.

On completion of an iteration

"*we will then bring users back in and we will ask them to go through typical usability testing model. Where you sit back and watch them use it*" [P6].

Because the developers take on the role of the UCDS, if a usability issue is discovered, the developer can deal with it immediately without passing it off to another team member. This also means that some developers implement features and are working in parallel to others who design the UI [P8].

The main difference between the Generalist and the Specialist approach is the roles that the developers need to practice. The data suggests that the working environment in the Generalist's approach was much less formal than that of the environment of the Specialists.

For instance, more than one UCDS that practiced the Specialist approach mentioned a sort of separation from the development team members. P9 remarked that the way the UCDS was accepted into a team environment depended on how they were introduced to a team. She mentioned that if she was introduced as a UCDS then she had to prove herself to the development and management team. Other UCD specialists [P1 and P5] mention that there was definitely a barrier of acceptance into the team. P5 referred to this barrier as the *"us and them"* perspective.

4.4 Refinement 3: The Facilitator

P2 and P4 followed a slight deviation to the Generalist and the Specialist, which we call the Facilitator. Their teams had a group member that had both formal UCD

training at the university level, and extensive software development and UCD experience. However, it is important to mention here that both P2 and P4 worked for very large multi-national companies, and both have extensive experience in their fields. The size of the company is important because of the politics that needed to be mitigated between the numerous customer groups and stakeholders. P4 remarked that their company dealt with very large enterprise projects and that there was going to be politics in projects of that size. P4 was there to aid in sorting out differences between different customers and to determine the UI features that would be designed and implemented in the next iteration. Although, P2 and P4 both had UCD and development expertise and were contributing their input into projects for which they were not directly designing the UI or developing. For this facilitation process, the team was not present and P4 acted as a bridge for delivering information back to the developers and UCDS. It may be the case that the size of the company dictated the Facilitator to be used between the UCD, developer, and customer groups.

This team member was both a Specialist and a Generalist. He was a Generalist in terms of having technical development skills as well as UCD skills. He also was a UCD specialist capable of performing this role expertly. The main difference between the Specialist and the General Specialist roles was the latter's team had more than one UCD specialist, and these were managed by the Specialist Generalist person. In the Specialist approach there was only one UCD person on the team with no development skills. Thus, for the purposes of this thesis we will call the team member with both formal UCD training and software development experiences a facilitator team member.

This approach was very similar to the Specialist and Generalist approaches in that it followed the same practices mentioned in the Initial Stage and the Iterative Stages. The main difference of this approach was in group membership roles. Both P2 and P4, at some points in the development cycle, acted as a liaison among all the different team members including the developers, the UCD team, and customers. For example, P2 stated that the UCD personnel on his team were "more or less divorced" from what the development team was developing. This meant that he was the only bridge between these two groups of team members. In other words, P2 was working with the UCD specialists to flush out the high level requirements. At almost the same time, P2 was also working with the development team at which point he acted as a bridge between the two groups by relating UI designs to development and implemented features back to UCD.

The main difference that this approach had was that the facilitator was not actually acting as the developer or the UCD specialist directly but mitigated between the two groups that never directly communicated.

4.5 Compromise: A Solution For Integrating The Methodologies

As discussed earlier in Chapter 2, agile methods and UCD have two very different approaches to developing software from an upfront resource allocation perspective. UCD tends to lean towards resource-heavy upfront research and testing prior to the code being written [17, 18, 19, 20]. Agile methods, on the other hand, champions minimizing upfront resource allocation in favor of more quickly getting working software into the hands of the customer [1, 4]. This spawns one of the key research questions driving this research and thesis: "How are these two methodologies being combined in industry given that they have very different upfront resource allocation techniques"? It is clear that these two

methodologies cannot coexist in the same development project without some form of compromise.

The agile practitioners needed to allocate upfront development resources for UCD and the UCD practitioners needed to scale back on the resources they typically required for upfront design. What was and was not compromised by both methodologies are key factors in the make up of the General Process Model and its three refinements.

4.6 Agile Compromises

Both methodologies needed to compromise in order to be combined. Agile practitioners needed to relax the notion of getting working software into the customer's hands as quickly as possible [4]. To do this they needed to accept that a portion of resources needed to be allocated for UCD practices in the upfront UI design. This meant that the time box for the predevelopment of the initial iteration (the iteration referred to as iteration 0) needed to be added to include resources for UCD practices. In other words, the Initial Stage had "*some measure of user research"* [P3] resources added to it. However, the Initial Stage was used to gather just enough information to get the project up and running. For example, P12 remarked that his team had an overall vision of the project but there was "*not much user research done upfront"*. P12's team were j*ust doing enough* [upfront research] *to get by"* to get the project started.

For some of the design teams, this Initial Stage had some flexibility.

"So typically when we start a project we go through what [company name] *calls a quick start. It's essentially like a project kick off, which can range anywhere from*

2 weeks to a month long of intensive sessions with different users. Bringing them in and doing high-level requirements gathering. [P6].

For others the Initial Stage timeline was a definite timeline. P12 also remarked that the time for their *"initial phase"* was averaging 2 weeks or under but not longer than 2 weeks.

P4 claimed their UCD team initially did a very general six month plan of the larger features set they wanted to deliver over that six month time period. They then selected and scaled the initial features set size back to fit the two week iteration time frame for the initial development design. This indicated that this team initially took a high level design and split it into more detailed smaller sets of features to fit the development iterations.

"So we sort of do a six month version right up front and then we have to scale it back and at the end of the initial iteration. We deliver the two week version" [P4].

Another participant recalled that they had a week of research and a week where the company they were working for supplied research material for them to draw from for design. These two weeks were predevelopment weeks and when asked if there was any UI development involved during these two weeks the participant replied *"not at all"* [P11]. This participant's Initial Stage lasted no longer than 2 weeks mainly because of budgetary constraints.

These are companies that are sort of on a shoe string budget and so they gather the user information. [P11].

The typical up front resource allocation was 2 weeks. However, the time allocated for up front UCD work varied slightly for some of the participant's teams. One participant remarked that their Initial Stage lasted two weeks [P8], while another stated theirs had lasted only a few days [P3].

"So, the idea that I found is the quickest way to get up and running is, is to create a set of personas. Start up with a well defined strategy phase which can be typically anything from a couple of days to a couple of weeks" [P11].

The longest reported time for upfront predevelopment resource allocation was six weeks.

"I had one and a half months time to acquire knowledge about users and build up the personas" [P10].

However, the data analysis showed that, for most participants, the Initial Stage on average lasted two weeks. Participants P2, P4, P5, P8, P10, P11, and P12 all stated that, typically, the Initial Stage was approximately 2 weeks whereas P3, P6 and P8 typically spent less than two weeks, P7 always spent more than 2 weeks, P9 spent six weeks, and P1 and P13 did not provide a timeline for their Initial Stages.

It was unclear what was responsible for the small differences in timelines between the different participants Initial Stages. However P3, P6, and P8 all shared the same approach to Agile UCD integration, the Generalist approach (discussed in section4.6). Their shared approach may have been the reason for the shortened Initial Stage. P2, P4, P5, P8, P10, P11, and P12 all shared either the Specialist refinement or the Facilitator

refinement and having a UCDS involved in their process may have been the reason for similar Initial Stage resource allocation.

4.6.1 What Was Not Compromised

Although Agile methods practitioners needed to extend the initial up front resource allocation to allow for UCD practice integration, only a short period of time was allocated in all cases compared to what typical UCD practices preferred. This meant that agile teams were willing to allocate some resources but not nearly the amount that UCD practices would normally require for a full upfront UCD study.

P10 claimed that he would have liked more time to complete more detailed upfront UCD research.

In all the cases the UCD folks had a limited timeframe to do their initial UCD study and design. As mentioned in the previous sections, there were groups that had some flexibility to carry out more detailed UCD studies

> *"Every once in a while we have the, you know, the ability to do the stuff* [usability research] *ourselves but It's not it's not a common case in all situations"* [P11].

4.6.2 When Agile is Integrated Into a Project

In all cases, agile methods was the software development methodology used by the participants in this study. Therefore, agile methods were present at the beginning and throughout the development lifecycle of each of the development projects that the participants worked on. This was, in part by, design as the goal of the study was to see how practitioners integrate both approaches. We didn't find a team that was using UCD

and then integrated agile approaches into their development process. This may be an artifact of the industry network of our group (which is better connected to the agile community than the UCD community).

4.7 UCD Compromises

UCD practitioners needed to compromise the notion of acquiring as much contextual information, design, and usability testing before any development could begin on the UI. The initial iteration was considerably shorter than that of a typical UCD driven project. This meant less time for contextual inquiry, UI design, and testing. It was inevitable that only a limited number of features could be designed in detail in the Initial Stage of development. This meant that UCD practitioners needed to give up a portion of the typical upfront resources. This was opposite to the agile methods compromise which required that methodology to add upfront resources.

For instance, the participants remarked that the upfront work of researching users and contextual design tended to be between two and six weeks which is much shorter than traditional UCD upfront resource usage [16, 17].

The amount of initial upfront design resource allocation for UCD was reduced in favor of developing the UI incrementally. This also reaffirms Sy's previously published model [6] discussed in the related work section.

> *"We had incremental usability studies, incremental development, incremental testing. Basically doing everything within an iteration"* [P12].

The iterative nature of the UCD design followed the agile methodology of iterations, and the output of the iteration typically was working code as well as incremental UI designs.

> *"Every 2 weeks there would be an iteration planning meeting where people would decide the work for the next couple of weeks* [for the designers]. *I didn't have any big picture design input, the kind of design input was something they* [the developers] *could fix in an iteration"* [P9].

4.7.1 What Was Not Compromised

Although the upfront resource allocation was cut dramatically for the initial iteration, the overall UCD process was not sacrificed. Only a fraction of the features could be designed and tested for usability in the time available for the initial iteration. This meant that UCD practices were applied to the features at different times in terms of iterations. The same UCD process used for the first iteration was also used in subsequent iterations. This meant instead of applying a majority of the UCD resources to the first iteration they were spread over subsequent iterations through out the projects.

> *"We have typical dev team members, we have business analyst. We also have a graphic design, interaction designer. We have human factors. ... UCD was with the majority of the project* "[P12].

The UCDS would typically find some issues with features in the UI design as testing was completed against the implemented features. Issues that required fixing may not fit into a single iteration and hence the UCDS would assign them to future iterations.

> *"I had issues I had found. So the testing basically helped me figure out, here's the things we need to fix first and then I would work with the product manager to make sure those things were in future iterations"* [P9].

Although the UCD upfront resources were cut in favor of iterative UCD design, all the features still were based on contextual inquiry, design, and had usability testing applied to them.

4.8 Timing of Integration

This section looks at the strategies for scheduling the UCD component into an agile software development project. The section discusses the strategies used and some of the problems and solutions that relate to integrating UCD into the agile development process from a timeline perspective.

4.8.1 When UCD is Integrated Into an Agile Project

The Related Studies, section 2.5, discusses different perspectives of bringing in UCD practitioners and practices into software projects. In that work, not much detail has been provided in terms of when agile methods integrate the UCD process into the overall development process. Although the papers in the related work section generally discuss if

the projects are new or existing, not much detail is provided about the different strategies used in terms of when UCD practitioners were added to projects.

During our research, we discovered that participants used two strategies for engaging UCD personnel in an agile team. For the purpose of this thesis, these two strategies will be referred to as the *Niche* strategy and the *End-to-End* strategy. The following sections discus these two strategies in detail.

4.8.1.1 The Niche Strategy

The first strategy that we observed for incorporating a UCD process into an agile project was the Niche strategy. For the purpose of this thesis, the definition of the Niche strategy is as follows; a strategy for *including a UCDS or UCD process into a discrete segment of a software project after that project has already begun. This strategy does not include the UCDS or the UCD process throughout the entire project lifecycle. This occurs after the project has been underway for some time and existing production UI code has previously been written for that project.* It is important to note that none of our participants were using the Niche strategy, although they had been exposed to it in the past and is therefore worth noting for comparison purposes.

This strategy is typically employed to deal with specific problematic usability areas of the UI software. This may be also employed when a project team member in a position of authority employs a UCD person in an attempt to improve their software development process as a whole.

> *"You are a trouble shooter. The guidance I'd been given by management was to focus on low hanging fruit in a sense. Fix what's easy to fix. I joined the team and*

the development team was already starting a release cycle. So within a month I had a whole snag list of all the issues that could be fixed" [P9].

One participant stated that projects that have no usability member on the team would eventually experience some sort of usability defect with their product after shipping. Eventually, that product may come back to the development team to have the specific problem(s) corrected. At this time, the UCDS may be employed in a Niche strategy to correct that particular defect or defects.

"Prior to product releases if you don't engage someone with interaction design skills and the product goes out, it's with issues. Usability issues that have to be addressed after the product has gone out to market" [P5].

At this time, the UCDS may be employed in a Niche strategy to correct that particular defect(s). They are brought in to address usability needs.

"So from the standpoint of understanding how the people really use the product there is nobody in the team that really represents that. Certainly development doesn't. The developers feel themselves capable of designing the UI. You can usually tell that this was designed by developers without the guidance of people like myself." [P5]

The Niche strategy is not aimed at correcting all the problems that may arise but to focus on more strategic usability problems.

"There is a lot of interaction design, UI design people, who work specifically on strategic stuff. So this application has this usability problem here that you have to work on. That's strategic in a sense that is very tactical and not certainly an end-to-end type of project" [P5].

One problem with this type of strategy is that often there may be more than the obvious defects that need to be corrected. P9 commented that when management brought her in to fix a specific problem, she found that there was a much larger list of usability issues to be corrected;

"When you are brought in just to solve one thing, you usually see four hundred others things that ought to be fixed" [P9].

For this particular project, the project manager (PM) was supportive and many of the issues were able to be resolved. The participant's reasoning for this was that there was an educational curve brought forth by her in order to demonstrate to the PM the need for these changes by including the PM in the usability testing process. However, the same participant also argued that if the PM on a project is not supportive of having the UCDS on the team, the opposite results may occur.

For instance, on one project business owners decided to employ a UCDS *"because they thought it would be a good idea to see how UCD would work in an existing product"*. On that project *"it was project management that was the prime source of resistance"* towards the UCDS. Resistance from project management included *"not*

listening, not willing to change any part of the plan in respect to what needed to be done to correct defects" [P9].

> *"They were not willing to do contextual inquiries. They were not willing to put in any testing. They were not open to any process of UCD"* [P9].

Another issue raised, in regards to the Niche strategy, was that employing UCD practices gives some team members the impression that the UCD process slows the development process down. The participants remarked that when a UCDS is brought into the development process and design issues are discovered, parts of the development process needed to stop. The claim was that this causes resistance from the development team members [P1, P5, P9]. In fact, one participant claimed it set up an *"us and them"* type of mentality with some developers.

> *"You do get Agilists who have worked successfully with usability people or design people. But frequently you get this sort of 'we know what we are doing. You get these usability people coming in and they slow us right down'"* [P5].

Another participant claimed that, because he had been brought in to the project in the middle, he felt he had to defend his *"opinions pretty strongly"* to have the team accept his design recommendations [P10]. These problems were among the reasons for participants favoring the second strategy discussed in the next section.

One solution offered by a number of UCDS was that other team members may need to be better educated in terms of UCD and the value of a UCD personnel on their team [P1, P5, P8, P9, and P10]. This was in keeping with the finding presented by

Gulliksen et al. in the related work section. Gulliksen et al. see education as a key factor in the successful integration of a UCD process into the development process [67].

One participant remarked that, in one project, the project managers were co-located and as a result they started to hear reports about UCDS team members on other teams. The managers of those teams realized that *"it's pretty handy to have this person around. You know she solves UCD stuff before it's actually seen"* [P9]. Thus, the value of a UCD person on the project became clear through watching and learning.

P9 also remarked that developers were rarely a source of resistance, which differed from what the previous participant stated about the project managers. What was clear was that there was resistance from more than one group. However, she claimed that there was still an instance of learning in that *"very quickly developers start to see, oh my God, this person makes my job easier"* [P9].

P1 claimed that once she had spent enough time designing a team's UI product, "*then you have a chance for them to appreciate what you are building"*.

This learning process is not limited to team members. Participants also stated that customer representatives thought that integrating UCD practices into the development process was a positive addition after witnessing the process first hand. The customer learning curve was reflected in our study data [P3, P6, and P8]. According to P6 customers/stakeholders from one project saw positive value in applying UCD in their projects. They also requested the use of some of the UCD practices prior to that being suggested by the team for the next iteration [P8]. This indicated that witnessing the value of UCD first hand added to their learning process. According to Gulliksen's findings *"all parties involved in the development of interactive technology, e.g. stakeholders,*

managers, users, etc. need to acquire a minimum level of awareness about HCI and usability" [67, pp 207-215]. In order to acquire this minimum level of awareness, it makes sense that education is key. This also supports Rosenbaum's notion that educating team members is a substantial portion of what is key to successful UCD integration into a software development process [68].

This is consistent with the suggestions provided by the participants in this study. Although our findings suggest that education may be an integral part towards better understanding of the value of UCDS in Agile software development, it is not limited only to the Niche strategy.

4.8.1.2 The End to End Strategy

Another UCD integration strategy found in our research was the "End-to-End" strategy. For the purposes of this thesis the End-to-End strategy is defined as; *the employment of a UCDS, and or a UCD process, from the beginning of an agile software development project and throughout the development lifecycle of that project.*

This means that the UCDS, or an acting UCDS, is present in the development process before the UI code is written and performs some form of upfront contextual inquiries and/or UCD testing and prototyping. Once the code development process begins, a UCDS or UCD process is present throughout the Iterative stages of the development process.

This does not necessarily mean that the first UCDS on the project is the same UCDS at the end of the project's development lifecycle. If projects span considerable time, there may be more than one UCDS involved in the process at different times. P10

stated that on previous projects that he had worked on, he was brought in from the beginning of the project as a UCDS. However, the project he was working on now, there was a UCDS involved from the beginning of the project but he was the third UCDS on this project.

Other participants also said that they had not been the original UCDS on a project but that the project had maintained a UCDS throughout the project's lifetime [P5, P9].

P2, P4, P11 and P12 all worked for large, multi-national IT companies that used the End-to-End strategy on a regular basis. Because these companies projects were so large and spread over longer periods of time, the members of the UCD team did change.

> There was UCD representation throughout the project from beginning to end.
>
> *"If you are talking about the entire project it was not just collocated it was spread over three different teams working in three different time zones, but the UCD was with the majority of the project."* [P12]

P6 also worked for a large multi-national company that was not as large as the companies that employed P2, P4 and P12. However, the company's size was substantial, spanning four continents. She commented that UCD "*is becoming much more prominent throughout the organization. So now all of our projects do incorporate it. They incorporate it holistically from beginning to end"* [P6].

One drawback with the End-to-End approach is that although the process allows for end to end testing, which is beneficial, it may be initially more costly than the Niche strategy. The End-to-End strategy requires that a UCDS be present throughout the project lifecycle, whereas the Niche strategy requires the UCDS to be present for a fraction of the time. This means that the resources required for the End-to-End strategy will be higher than those required for the Niche strategy. This may explain why larger companies are able to employ UCD practices in an End-to-End strategy. The cost of integrating UCD practices can be substantial. On average, UCD constitutes 19% of software development budgets [70]. Perhaps for some smaller organizations these costs may be prohibitive in terms of End-to-End integration. On the other hand, the Niche strategy does provide at least some UCD benefits for companies or projects with fewer resources.

As mentioned above, P2 and P4 both worked with very large companies. Both of these companies employed End-to-End strategies on the projects P2 and P4 had worked on. They were both satisfied that the methods being used were successful. P4 remarked that his company had a large resource pool of research for what practices had worked in the past and that the teams he worked on relied on this information during their projects.

P5 stated that she had worked on both Niche and End-to-End strategies but that she preferred working on a large-scale project from the beginning to the end. Her reason for this preference was that "you have to have something that is successful for the moment as well as for future." She also stated that in a Niche approach it is very tactical and the UCDS does not see the end result of the project.

P9 also favored the End-to-End strategy in that she found it less frustrating in terms of limited time and resources. This participant commented that it is very frustrating

when you are employed to fix a specific problem and you see that there are many other problems that require fixing but you are not allowed to do anything about them. P3 and P6 were also very much in favor of the End-to-End approach. As mentioned above, their company was very much behind UCD practices and that they were employing them "from start to finish of their projects".

4.9 Chapter Summary

This chapter presented the findings of our study. First, the section introduced the General Process Model which is a high level overview of how agile and UCD methodologies are being integrated. Next, it briefly discussed the tandem like development process that allowed integration of both methodologies. This was followed by the three refinements to the AUGPM, The Specialist, The Generalist and the Facilitator refinements. The chapter then discussed the compromises that both Agile and UCD processes needed to make in order to coexist on the same project. Finally, two strategies were introduced in terms of integrating a UCDS or into an agile project. These were the Niche strategy and the End-to-End strategy.

5.0 Conclusion

The following chapter provides a conclusion for our study. It contains a brief discussion of our findings and revisits the existing work in order to establish that our findings have commonalities with previous work that can be compared against numerous software projects and teams. Next, this chapter discusses some of the limitations of the study and provides suggestions for possible ways to strengthen future studies. Finally, the section concludes with a discussion of possible future work.

5.1 Discussion

Our study set out to investigate how agile methods are currently being integrated with UCD practices and in turn reconfirm (or refute) existing work in this area of study. We first started by examining related work. We found that, although these studies provided valuable insight for our work, a broader empirical basis is needed to make more general conclusions in terms of the integration of these methodologies.

With the exception of Ferreira's work, the previous studies that pertained directly to Agile and UCD integration centered on one team or one company. There was no strong evidence that these approaches were general enough for use in different companies or on multiple teams. Ferreira's study [9], on the other hand, did span 4 projects and 4 different teams. The process descriptions in that study were not detailed in terms of the steps in their processes. Our study examined both the steps in the process and the process over 13 different teams in 12 different companies to find commonalities shared for integrating agile methods with UCD presented in Section 4.0. All our participant's teams followed similar processes to that of Sy's. This confirmed the model originally presented

by Sy [6] and suggests that this integration approach is not only being used but could be more widely applicable to other teams and projects. This was outlined in the AUGPM in Section 4.1. In fact, three of the participants were employed at multinational companies that are using this approach on multiple teams in different parts of the world [P2, P3, and P4].

The participants' teams, however, did have some differences in terms of team roles and responsibilities. We also presented different refinements to the AUGPM regarding who performed the UCD-oriented activities in a team in terms of their roles. These refinements were: the Specialist, the Generalist, and the Facilitator, Sections 4.2, 4.3, and 4.4 respectively. These refinements present the answer to another research question: what are the roles that interplay directly with the combination of these two methodologies?

We also found that these refinements were similar to approaches used in previous work.

Sy's previously published work, Adapting Usability Investigations For Agile User-Centered Design, was, by far, the most detailed paper in terms of UCD Agile integration. It closely resembled our AUGPM, presented in Section 4.1, and more specifically, the Specialist refinement presented in Section 4.2.

Sy's approach had UCDS members on their team that gathered the upfront data in Iteration 0, much like in the Specialist refinement. Once Iteration 0 was completed, the developers and UCDS worked in parallel during the cycles that followed [6] as described in our Tandem definition Section 4.1.1. This approach closely matched the AUGPM and the Specialist refinement of our study in a number of ways. First, there was an Iteration

Zero for conducting contextual inquiry and prototyping by the UCDS in the same manner as the Specialist approach. Second, Iteration Zero was completed before any implementation of the UI code took place. Finally, Sy's process included team members that were specific to the UCDS role which closely resembled our Specialist refinement.

The Generalist refinement, presented in Section 4.3, was similar to Patton's process [7]. On his team, there was no formal UCDS. The development team was expected to contribute to the UI design acting as the UCDSs. Therefore, the development team was responsible for both UI design and the implementation of that design. Each developer was expected to perform heuristic evaluations on any new feature that was introduced into the UI much like a UCDS would during the course of their evaluation. This was similar to the usability inspection activities described in Section 4.3. The team members on Patton's team actually designed the features and their placement on the UI similar to our Generalist refinement [7].

In Ferreira's four case studies, the UI design and development process aspects were not as detailed as the other related work; however the roles of the team members were. Her cases, one, two and four, included one team member that was an actual UCDS. This resembled the Specialist refinement in terms of having at least one UCDS on the team. Her case three resembled the Generalist approach in that the team member acting as the UCDS was also a developer who had an interest in using UCD methodologies in the software development process. There was no clear indication that the UI designer was formally trained in UCD practices. She only states that UCD UI design was their interest [9].

All of the related work discussed above did resemble one of the three approaches that we saw in our data. Moreover, the process refinements in our data as well as the processes in related work are consistent and complete (in as much as no other process models were described yet). The empirical data gathered by others and in the study presented here indicates that the common approach outlined above can be – and is – used by different companies as well as by different teams.

One more similarity is that all of the empirical work so far did mention a degree of success when implementing agile methods together with UCD practices. We also found that all but two of our participants suggested that by combining these two methodologies there was value added to their development process. The other two participants made no comment as to the value added by combining these two methodologies.

As mentioned in the introduction, a key difference between agile methods and UCD is the upfront resource allocation for planning and developing UIs. Our findings introduced in Section 4.5 show that these differences were overcome in all the approaches in the same way; compromise. This answered a third one of the key research questions put forth by this study: "how are these two methodologies combined given that they have very different upfront resource allocation techniques"?

Some upfront effort was invested in the Initial Stage by all participants. But this upfront design effort was rather limited compared to traditional UCD processes and more resource heavy than traditionally allocated by agile teams. With this fairly short timeframe, lengthy upfront research required by traditional UCD practices could not be performed in much detail. This meant that smaller sets of UI features were gathered prior

to implementation. It also meant that no UI code could be written by development until those features were designed. This definitely had an effect on both methodologies in terms of their original approach or process. This speaks to the forth of our research questions: What effect does the integration of these two methodologies have on their original approaches or processes?

Another aspect of our findings was the strategies for bringing the UCD process into an agile project. As previously mentioned, all the participants projects were agile from the beginning of their development lifecycle and therefore no findings relating to the impact of integrating agile into an existing UCD process were found. On the other hand, there were significant findings for when a UCD group member was brought into an agile project. The two strategies that were evident were the End-to-End and the Niche strategies presented in Section 4.8. This speaks to another of the research questions: what strategies are being used to incorporate an interaction design process into an agile software project?

Previous work points to the End-to-End being more effective than the Niche strategy among the UCD professionals community [67, 71]. Our findings also suggest that the participants favored the End-to-End approach.

There were other key factors that our participants felt were important to the success of UCD integration into the process. These were: support from upper management, support from project management, and support from users, as well as acceptance of UCD practices from developers put forth in Section 4.8. These key factors confirmed the work by Gulliksen et al. [67].

For example, P1 stated that support came from developers and management alike once they realized the contribution of the UCDS actually made a better product. P3 stated that after the first implementation of UCD into their process the managers and customer/users asked for the process to be included in their next project. One participant recounted a situation where a PM was not in favor of a UCDS being added to the process but that the support did come from upper management and that was of great help [P9]. Other participants that were working with large corporations had support based on previous projects and corporate acceptance of their previous success using UCDS throughout the development process [P2, P4, P11, and P12].

5.2 Limitations of the Study

In this section we look at the limitations of this study.

This study is based on interviews with the participants. It may have been beneficial to have observed the participants and their teams in their working environment in order to gain the context of the participants experience first hand. This may, or may not, have provided a different perspective in terms of their process of UCD Agile methods integration.

Another facet of this study was that none of the participants worked on the same team. This meant that there was a developer, a UCDS (or acting UCDS), or a Facilitator interviewed from different teams but not from the same team. It may have been beneficial to interview multiple individuals in two or three of these different roles on each team. This may have provided further insight on the integration process from the different interacting roles on the same team.

Finally, all the participants had experience with UCD being integrated into agile methods and not agile methods being integrated into a more traditional process with UCD in place. It is difficult to determine if having both agile methods integrating UCD and UCD integrating agile methods into their methodologies would have changed or further reinforced our conclusions.out

5.3 Future Work

This study uncovered some interesting facts in terms of the way agile methods and User-Centered Design are being integrated in industry. However, there were new questions that emerged from that study as a result.

For example, all but two participants commented that they thought that the integration of agile methods and UCD into their process was successful. An important question that may be worth investigating is "Why did those participants feel that their process was successful"? Another question along the same lines might be "How did you measure success in your process to determine it was successful"?

Another theme that emerged from the data was the social ramifications of when a UCDS joins a software project team. Comments like there was an "us and them" mentality that surfaced with the developers towards the UCDS [P5], or "it took a while before the developers realized that we made their life easier" [P9]. Other important socially motivated questions that may be worth investigating might be: "Which is an effective way to integrate a UCDS into an agile software development project to minimize social out casting and maximize team gelling?" "Is the use of the name Usability Specialist a contributing factor for team member integration resistance or is it just a question of lack of knowledge on the other team member's behalf?"

Finally, another key question that may prove to be valuable might be "what is the optimal time, in terms of months, weeks, or days for upfront work to be completed by the UCDS before development begins"?

6.0 Bibliography

1. Fowler M.: The New Methodology, July 2005 [Online]. Available: http://www.martinfowler.com/articles/newMethodology.html. [Accessed Feb. 12 2008]
2. Royce W.: Managing the Development of Large Software Systems. Proceedings, In WESCON, 1970 pp 328-338 IEEE Press, 1970.
3. Marcus A.: User-Experience Planning For Corporate Success, Interactions, volume 11, issue 3, pp. 24-27, ACM Press, 2004
4. Principles Behind the Agile Manifesto (ND) [Online]. Available: http://agilemanifesto.org/principles.html [Accessed April 24, 2008].
5. Mezsaros G., Aston J.: Adding Usability Testing to an Agile Project, In Agile 2006, pp. 289-295, ACM Press, 2006.
6. Sy D. Adapting Usability Investigations for Agile User-Centered Design Journal of Usability Studies ,volume. 2, issue 3, pp 112-132, ACM Press, 2002,
7. Patton J,:. Hitting the Target: Adding Interaction Design to Agile Software Development, In OOPSLA, pp 1-7. ACM Press, 2002.
8. Maurer F., and McInerney P., UCD in Agile Projects: Dream Team or Odd couple Interactions, volume 12, issue 6, pp 19-23, ACM Press, 2005.
9. Ferreria J., Nobel J., Biddel R.: Interaction Designers on eXtreme Programming Teams: Two case Studies From the Real World, *Proceedings of the Fifth New Zealnad Computer Science Research Student Conference*, Hamilton New Zealand, 2007.

10. M Maurer F., Melnik G.: Agile Methods: Crossing the Chasm, In ICSE 2007, pp. 176-177, ACM Press, 2007.
11. Abrahamsson P., Warsta J., Siponen M., and Ronkainen J., New Direcetions on Agile Methods: A Comparative Analysis, In ICSE 2003, pp 244-254, ACM Press, 2003.
12. Highsmith J.: History: The Agile Manifesto 2001 [Online]. Available: http://agilemanifesto.org/history.html [Accessed April 25, 2008].
13. Conboy K., and Fitzgerald B., Towards a Conceptual Framework of Agile Methods: A Study of Agility in Different Disciplines, Proceedings of 2004 workshop on interdisciplinary software engineering research, pp 37-44, ACM Press, 2004.
14. Maurer F., and Melnik G., Agile Methods: Moving Towards the Mainstream of the Software Industry, In ICSE, 2006 pp. 1057-1058, ACM Press, 2006
15. Bacon E.: Defining Interaction Design, Interactions, volume 12, issue 3, pp. 34-35, ACM Press, 2005.
16. Nielsen J. *Usability Engineering,* San Diego California: Academic Press, 1993, pp. 23-27.
17. Preece J., Rogers Y., and Sharp H., *Interaction Design: Beyond Human-Computer Interaction,* John Wiley & Sons: New York NY 2002, pp. 8-9,
18. Baecker R., Grudin J., Buxton W., and Greenberg S. *Human-Computer Interaction: Toward the Year 2000,* 2cd ed., San Francisco: Kaufman Publishing 1995, pp. 95-106.

19. Preece J., Rogers Y., Sharp H.: "What is Interaction Design" in *Interaction Design: Beyond Human-Computer Interaction* John Wiley & Sons: New York NY 2002, pp. 8-9.
20. Barnum I, *Usability Testing and Research,* 1st ed., Campion, Ed. NY New York: Longman Publications 2002, pp. 187-188.
21. Mao J., Vredenburg K. Smith P, Carey T: User-Centered Design Methods in Practice: A Survey of the State of the Art, *Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research* pp. 12-ff, IBM Press, 2001.
22. Beck K., Extreme Programming Explained, Addison-Wesley: Upper Saddle River NJ. 2000.
23. Beck K. and Fowler M., Planning Extreme Programming, Addison-Wesley: Upper Saddle River NJ. 2001.
24. Cooper A., The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity. Pearson Education: Don Mills ON. 1999.
25. Chaos Report Standish Group Report 1995. [Online]. Available: http://wwwlib.murdoch.edu.au/find/citation/ieee.html#books. [Accessed May 8, 2008].
26. Cunningham W.: Manifesto for Agile Software Development, 2001. [Online]. Available: http://agilemanifesto.org/. [Accessed: May 8, 2008].
27. Armitage A.: Are Agile Methods Good for Design, Interactions, volume 11, issue 1, pp. 14-23, 2004,

28. Fowler M.: Is Design Dead? [Online]. Available: http://www.martinfowler.com/articles/designDead.html [Accessed April 3, 2008]
29. Hartman D. Interview: Jim Johnson of the Standish Group, 2006. Available: http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS. [Accessed: May 10, 2008].
30. Maurer F., Melnik G. What You Always Wanted to Know About Agile Methods But Did Not Ask. In ICSE 2006, pp. 731-731, ACM Press, 2006.
31. Beck K., Andres C.: Extreme Programming Explained: Embrace Change, 2 ed. Addison-Wesley, Boston, MA, 2004.
32. Merriam-Webster Online Dictionary. Available: http://www.merriamwebster.com/dictionary/scrum [Accessed: May 16, 2008].
33. Takeuchi H., Nonaka I.: The New Product Development Game, Harvard Business Review, pp 137-146, 2005
34. Schwaber K., Beedle M.: Agile Software development with Scrum. Upper Saddle River NJ : Prentice Hall, 2002.
35. Schwaber K.: Agile Project Management With Scrum, Redmond Washington: Microsoft Press, 2004.
36. Schwaber K. *Explanation of Scrum* Available: http://www.controlchaos.com/download/Scrum%20Explanaion.pdf [Accessed: May 16, 2008].
37. Schwaber K.: Scrum Rules, [Online] Available: http://www.chaoscontrol.com/old-site/rules.htm [Accessed: May 16, 2008].

38. Schwaber K.: Scrum Alliance: No Applause Please, 2006, [Online] Available: http://scrumalliance.org/articles/31-no-appluase-please [Accessed: May 18, 2008]
39. Bacon E.: Defining Interaction Design, Interactions, volume 12, issue 3, pp. 34-35, ACM Press, 2005.
40. Norman D. The Design of Everyday Things, MIT Press: New York, 2002
41. Law F., Kort J., Roto V., Hassebzahl M., Vermeeren A.: Towards a Shared Definition of User Experience, In CHI 2008, pp. 2395-2398, ACM Press, 2008.
42. Alben L.: Quality of Experience: Defining the Criteria for Effective Interaction Design, Interactions, volume 3, issue 3, pp. 11-15, ACM Press, 1996.
43. Hassenzahl M., Tractinski N.: User Experience – A Research Agenda, Behavior and Information Technology, volume 25, issue 2, March April 2006 pp. 19-97.
44. Wright P., McCarthy J.: Empathy and Experience, In CHI 2008, pp. 637-646, ACM Press, 2008
45. Hewett T., Baeker R., Card S., Carey T., Gasen J., Mantei M., Perlman G., Strong G., Verplank W.: ACM SIGHCI Curriula for Human-Computer Interaction. [Online] Available: http://www.sigchi.org/cdg/cdg2.html#2_1 [Accessed: May 19, 2008]
46. Juristo N., Ferre X,: How to Integrate Usability into the Software Development Process, In ICSE 2006, pp. 1079-1080, ACM Press, 2006
47. Maguire M.,: A 15 Year Path of Usability Development in Europe, In CHI 2000, pp. 195-196, ACM Press, 2000
48. Lee J.,: Embracing Agile Development of Usable Software Systems, In CHI 2006, pp. 1767-1770, ACM Press, 2006.

49. Vredenburg K., Mao J., Smith W., Carey T. “A Survey of User-Centered design Practice” In SIGCHI 2002, pp. 471-478, ACM Press, 2002
50. Mao J., Vredenburg K., Smith P., Carey T.: User-Centered Design Methods in Practice: A Survey of the State of the Art, In CASCON 2001, pp.12-ff, ACM Press, 2001.
51. Mao J., Vredenburg K., Smith P., Carey T.: The State of User-centered Design, Communications of the ACM, volume 48, number. 3, pp. 105-109, IBM Press, 2005
52. Vredenburg K.,: Increase Ease of Use: Emphasizing Organizational Transformation, Process Integration, and Method Optimization, Communications of the ACM, volume 42, number.5, pp.67-71, ACM Press, 1999
53. Raven M., Flanders N.: Using Contextual Inquiry to Learn about Your Audiences, ACM SIGDOC Asterisk Journal of Computer Documentation, volume 20, number. 1, pp. 1-13, ACM Press, 1996.
54. Constantine L. “Beyond User-Centered Design and User Experience: Designing for User Performance”, Cutter IT Journal, volume 17, number 2, [Online] Available: http://www.foruse.com/articles/beyond.pdf [Accessed: May 24, 2008]
55. Cooper A.,: About Face 3 –The Essentials in Interaction Designs,, New Jersey: Wiley Publishing, 2007
56. Nielsen J.: Finding Usability Problems Through Heuristic Evaluation, In CHI 1992, pp. 373-380, ACM Press, 1992.
57. Mack R.: Nielsen J.: Usability Inspection Methods, SIGICHI Bulletin volume 25, issue 1, pp. 28-33, ACM Press, 1993,

58. Ferreira J.: Interaction Design and Agile Development: A real-World Perspective, M.S. thesis, Victoria University of Wellington, Wellington New Zealand , pp. 61-117, 2007.

59. Nelson E.: Extreme Programming vs. Interaction Design, [Online] Available: http://web.archive.org/web/200060613184932/www.fawcette.com/interviews/beck_cooper/default.asp/ [Accessed: May 20, 2008].

60. Constantine L.: Process Agility and Software Usability: Toward Lightweight Usage-Centered Design, [Online] Available: http://www.foruse.com/articles/agiledesign.pdf [Accessed: May 24, 2008].

61. Powel R.: Recent Trends in Research: A Methodological Essay, Library and Information Science Research, volume 21, issue 1, 1999, pp. 91-119

62. Gorman J. and Clayton P.: Qualitative Research for the Information Professional: A practical Handbook, 2nd ed., London: Facet Publishing, 2005

63. Abusabha R. and Worlfel M.: Qualitative vs. Quantitative Methods: Two Opposites That Make a Perfect Match: Journal of the American Dietetic Association, volume 104, issue 5, pp. 566-569, May 2003, [Online], Available: http://find.galegroup.com.ezproxy.lib.ucalgary.ca/itx/retrieve.do?contentSet=IAC-Documents&resultListType=RESULT_LIST&qrySerId=Locale%28en%2C%2C%29%3AFQE%3D%28KE%2CNone%2C37%29qualitative+vs.+quantitative+research%24&sgHitCountType=None&inPS=true&sort=DateDescend&searchType=BasicSearchForm&tabID=T002&prodId=AONE&searchId=R1&currentPosition=1&userGroupName=ucalgary&docId=A101796918&docType=IAC [Accessed: June 5, 2008]

64. John M., Maurer F., and Bjornar T.: Human and Social Factors of Software Engineering – Workshop Summary, ACM SIGSOFT Software Engineering Notes, volume 30, issue. 4, pp. 1-6, ACM Press, 2005
65. Urquhart C.: An Encounter with Grounded Theory: Talking the Practical and Philosophical Issues, In Qualitative Research in IS: Issues and Trends, pp. 115 Hershey PA: Idea Group Publishing, 2001.
66. Strauss A. and Corbin J.: Basics of Qualitative Research, 2nd ed., Thousand Oaks CA: Sage Publications, 1998.
67. Gulliksen J., Boivie I., Persson J., Hektor A., Heurf L.: Making a Difference: A Survey of the Usability Professionals in Sweden, In NordiCHI 2004, pp. 207-215, ACM Press, 2004.
68. Rosenbaum S.: A Toolkit for Strategic Usability: Results from Workshops, Panels, and Surveys, In SIGCHI 2000, pp. 337-344, ACM Press, 2000
69. Venturi J., Troost J.,:Survey on the UCD Integration in the Industry, In NordiCHI 2004, pp. 449-452, ACM Press, 2004
70. Vredenburg K., Mao P., Smith P., Carey T.: A Survey of User-Centered Design Practice, In CHI 2002, pp. 471-478, ACM Press, 2002.
71. Bruno V., Dick M.: Making Usability Work in Industry: An Australian Practitioners Perspective, OzCHI 2007, pp. 261-264, ACM Press, 2007.
72. Nielsen J.: Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier, Academic Press, pp. 245-272, 1994.

73. Fox D., Sillito J., Maurer F.: Agile Methods and User-Centered Design: How These Two Methodologies are Being Successfully Integrated in Industry, In Agile 2008, pp. 63-72 , IEEE Press, 2008

74. Life With Software Engineering (ND) [Online] Available From: http://life-with-software-engineering.blogspot.com/ [Accessed January 3, 2010]

75. Boehm B.: A Spiral Model of Software Development and Enhancement, volume 21, issue 5, pp 30-37, Computer, May 1988

Appendix A – Study Questions

The following are the four main open ended questions followed by questions that arose as a result of those main questions.

- Can you tell me about your background in terms of employment, or project types you have worked on, and education/training?
    - What is your position on the team?
    - What is your role on your team?
    - What is your educational background?
    - What types of projects have you and are you working on e.g. Agile?
    - Do you work with a usability specialist when you are developing say the UI?
    - How long have you been doing this?
    - What can you tell me about the company you work for?
- What are the activities surrounding your software development approach?
    - Where does your process start?
    - What happens up front?
    - Who are the members of your team?
    - Who are the members of the team are you readily interacting with?
    - Are you mitigating between UCD people and developers?
    - Are you going into a work place or where are you drawing on you contextual research?
    - Who are at the planning meetings?

- Is your team collocated?
- Do you user real users in your UCD testing?
- How do you go about testing?
- What tools do you use for prototyping?
- What are the roles on your team?

- Can you describe how the activities are put together to fit into your software development process?
  - How you go through a typical the iteration?
  - In a typical iteration is the UCD team staying ahead of a development cycle in terms of UI testing and design?
  - Who are the members of the planning meeting?
  - Does the UCD get passed before the end of the sprint for some sort of testing or verification, or is it all just done at the end?
  - During the initial phase there is no development taking place?
  - So how long would this start up phase as the initial phase last?
  - Do you think that high fidelity prototyping restricts your teams design or development?
  - Are there tools out there that you think might help your overall process in terms of UCD and Agile development?
  - Is development going on at the same time as UCD design and testing?
  - Do you feel Agile methods is a successful development process?
  - Do you feel adding UCD to the Agile methods process is effective?
  - At what point do you the usability people come in?

- When does the UCD design get passed to development?
- How long are typical iterations.
- How much time do you spend before development begins?
- Is there a UCD person(s) throughout the development lifecycle?
- How often do you communicate with the UCD folks/development folks?

- Can you tell me how the evolution of inducting Agile methods/ UCD into your process transpired?
  - When you brought in UCD persons is there resistance either from management, developers, or UCD people?
  - Are there issues up front bringing in UCD processes?
  - Did bringing in UCD impact everybody in a positive perspective?
  - Do you feel that Agile and UCD fit together well?
  - Did you bring in a UCD person from the beginning of the project?
  - What did you do to overcome the obstacles of bringing UCD into your process?
  - Do you feel that the time given to predevelopment was sufficient?
  - Are you finding that your application development is more successful using these two processes together?
  - Do you intend on keeping your current practices in your development process?
  - Is there a tool for development/prototyping that is, or is not existing that you think would be helpful in your software development process?

Appendix B – Open Codes

The following are the open codes with example of the text the codes were applied to.

Table 4: Open codes applied to transcription data examples.

PAR-TICI-PANT	OPEN CODE	EXAMPLE
P6	Addressing change the changing the process of development	Seeing projects going over budget, seeing clients frustrated with seeing how much they were spending and receiving such little value . So um decided to focus my career on Agile and went in search of [company name] and ah after a long courtship was ah I started with [company name] a little over a year ago.
P3	Adopting a methodology for their needs	Agile as a classic methodologies or classic practices that have similar characteristics and within Agile development you will find extreme programming and scrum and feature driven development and all these specific recipes and within Agile development you will find that no one follows any 1 recipe. Rather they pick and chose and they blend and they create an ad hoc or a situation specific methodology.
P8	Individuals process comparison to an Agile author's	So that's just an accommodation that we've already had to deal with reality the fact that rarely do you have a single person conducting meetings with all the users, making all the decisions in some fashion. Which is ideal for the Kent Beck model for XP.
P8	Concerned about UCD process from an Agile perspective	But for areas that [Agile process] doesn't work very well is UI. And that's partly because users don't like you changing the interface but also the tools for UI development don't seem to be nearly as good. A lot of them are really clunky and when you start moving things around you lose the code
P9	An Agile Model component being	Pair programming was the other part of the Agile development environment that I worked in.

	used in a participant's process	
P4	Agile requirements gathering	We try to get everyone on the phone and walk them through the wire frames. You know let them explore, let them call out well this one doesn't work for me. Well why doesn't this one work for you? Is it just one of your personal preferences? Or is this a fundamental business reason why this one doesn't work for you?
P4	Agile requirements gathering issue – overcoming it this way	So, so we often have 2 versions of the wire frames. 1 version of the wire frame has all these annotations with all the politics and all the stuff for the stakeholders. Then we have another version of the wire frame that has the technical data. That says here's how to build this and here's how to build that. And the developers don't care about the politics and the stakeholders don't care about the development details.
P3	Agile term - Author specific	Now today it is ok to estimate a story from between a half day to about 2 and a half days. Um and that's ok. So that today is a user story and these big things are what Mike Cohen calls an epic.
P3	Agile testing perspective the developers used.	We didn't do usability testing we done this kind of continuous self inspection
P7	This Agile term is used interchangeably with UCD	… things called iterations
P5	Alliances	Sometimes those [team members and marketing stakeholders] are our best sort of alliances because they have a lot of information about what customers are saying about the product.
P11	No available information about a product to be developed	A lot of it has to do with that but you know of course sometimes it's, it's a product that no one else really has anything like that so this is a little more difficult.
P11	What helps with the big picture (overall vision)	The other thing in retrospect is that you know at the same time we should have adopted more of you know the UCD type of I guess more, more from a traditional stand point. More of a big design up front but kind of just enough upfront that it would help us really think out the

		developing system thinking types of things. Um that would have been helpful that came back to bite us later.
P12	Participant concerned about building the right product	I think I think that what it resulted in was building the right portions of the product. Um and getting the product to the market faster.
P8	Cart before the horse – putting product technology before the user	So it's really the analogy of putting the technology cart before the horse. If you're thinking serious from the technology perspective of we could build some software were not really thinking about the user.
P7	Communication throughout with the user	So we bring users in at the requirements level.
P11	Communication with the developers and UCD	. there was continuous feedback between those 2 groups um during the entire sprint
P10	UCD communicated with the developers this often	By email I talk with them about 2 times a week. By phone like 2 times that amount. And some times we have used instant messaging like chatting. But a few times when they have come to our side and we have talked, I have talked with them it's like, it's like what we can accomplish in 2 weeks by emailing and phone by phoning in 2 days we can do that multiplied by 10.
P2	Communication between the developers and UCD/bridge UCD	so it [conversations between UCD and development] would be more of a kind of morph into. That would be more of a, just a kind of conversation and that conversation would happen in real time as opposed to being tossed back and forth. does that makes sense
P1	Communication issue.	Well sometimes it changes and sometimes it's technical you know it might be in a domain that we're not at all familiar with.
P1	Communication issue between developers and UCD people	And sometimes you're in a situation where the … sometimes you're in a situation where the customer and the developers are very far apart from each other.
P9	Communicating with the product manager	So you would talk to the product manager. Because the product manager had to be convinced that this was important from a business point of view
P11	Company size	What is interesting about [company name] is a very large corporation
P12	Compromising methodologies	You had these UCD people coming in and they have to learn to adjust to not doing everything upfront.
P4	Consequences of evolving	Well what we're pushing hard for is to get our executives to really buy into the philosophy that this is a

	methodology	project and you know everybody needs to work together as a team and don't give us so much work to do. Cause if, if we are left to our own devices we will do just fine.
P3	Consequences using Agile incorrectly	We are talking about a user story that may take 75 man days and ideal days are these ah weird squiggly word that XP uses. Which means, there is some load factor applied to that. A common load factor for that is 3. So we are talking about a user story that might have been estimated in 25 days. Today, in 2007, if estimated a user story in 25 days you'd be booed off the agile stage. Um you would be sent straight to hell [ha ha ha]. Um straight to waterfall hell and you would be forced to stand under the waterfall until you, you learned your lesson.
P11	Contextual inquiry - research	Typically in the second week what's happening is we're doing the research, I'm doing conceptual models too I should mention that.
P10	Contextual inquiry	So first when the projects didn't start [designing] I had 1 and a half months time to inquire of the knowledge about users
P11	Customer role	Usually there the executive staff. We're dealing a lot with the CEOS, the CTOs and all those COs..
P1	Customer giving requirements to designer	If you can do it in this time frame we want it.
P2	Customer voicing requirements	And I met with the customer came in and even before I could get to any of that they said to me "look we're going to take you thru the vision of you know, things we want to achieve for the next period of time".
P6	Customer understanding	You know it 's you know it's an interesting thing with us because of the type of clients we have. It's not that they don't want it, there's 2 aspects of it. First it is understands exactly what it means.
P5	Dependence work – dependencies or non-dependencies in a team environment	In some cases there is a lot of dependencies and generally in an agency there is sort of a tighter timeline so there isn't a lot of time to sit there. In my job now you have a lot more time and you also work more independently. You are not tightly coupled with a team the way you are with an agency.
P5	Design do's and don'ts	So they can, well they feel themselves sometimes capable of designing that stuff, you can usually tell that this was designed by developers without sort a the guidance of people like myself who, who can soften the edges sometimes.
P9	Design is passed from the UCD team to the	And then to deliver the designs and the interaction details specs at the beginning of the iteration.

	developers	
P4	Design goes to business	But eventually we will give it [the design] to a larger team that that's outside of our team that would be focused on the business side is what you might call it.
P4	Design goes to the UCD team	Usually what happens is eventually we get to an agreement and then it goes into a more interaction design and the visual design stage. Where either the wire frames get flushed out in more detail where we will actually pay attention to some of the spacing. So it will be still be like you know like a higher fidelity wire frame.
P4	The design or application goes to the customer or acting customer for approval.	We see the pros and cons of A and B so we are actually going to take A and B to different groups outside the team. They would be different stakeholders.
P12	Developers being defensive to other team members	[I have seen this] not necessarily here but I would say in my career previous to being here I had observed that.
P9	Developers testing step happens this way	You know so there would be 2 or 3 days of development and then testing would start you know a couple days later and then it would basically… they would clean up what ever testing found with in the iteration.
P4	Documentation	But we are finding that it is pretty easy to document it for the developers. But then we also have to do another kind of documentation for the stakeholders.
P12	Doing things differently than in the past ad reflecting on it	But you know in retrospect it the thing that was it was better than we started out and this mostly partial... mostly my fault. When we started out I thought you can do the design and then just build the entire design but in retrospect it is easy to fail because everything is based on a development mind set. It is easy to miss the big picture. the places where we had UCD you know the people we would group into that UCD bucket; graphic designers IDs, and human factor engineers, and usability engineers as well. um I think that for them to really excel and thrive they needed to be able to think at a systems level thinking. and you know in retrospect
P4	Doing a methodology or practice without knowing	I think that it was all of the standard reasons. It was just compounded because in the web you are kind of forced to do that anyway. It's because you got your stuff done so fast or you can't compete. So it's, it's like a no brainier when you are working on the web. Even if you don't call it Agile most of the teams I've worked with do some version of it Just because it's baked into your blood if you are doing web work.

P9	End-to-end strategy – having a UCD person involved throughout the entire project	Well I lean quite heavily towards end to end because in my experience, when you are brought in you know just to solve one thing you usually see 400 other things that ought to be fixed. And it is extraordinarily frustrating as the user usability person to then you know I don't know to populate 1 dialog or do something small
P10	End user – their role/ description	I you ask me if the role of the end user shouldn't be too big when designing design because users/customers can't focus and on focus on wrong things. They start to focus on key things and not acting out enough of the interaction behind the in the product; interaction behind the product.
P12	Found the project/methodology successful as a result of this practice/variable	This [UCD inclusion] allowed us to build things much faster um and also to build the right thing.
P6	Found the project/methodology unsuccessful as a result of this practice/variable	My background on a number, couple of waterfall projects was up front doing user interviews. And um over the course of 5 years I've probably worked on I'll say 10 different projects where I didn't see a single one of them complete development because they ran out of money
P2	The team member found this to be a positive aspect of the methodology they are using	And I would say of all the Agile practices the 2 most valuable are the concept of the short time-boxed releases and having the customer as part of the team. So those are the, those are the things that create the greatest value I think.
P5	Gaps understanding their process	Well I am still trying to figure that out actually [their process] I have been here I don't know about 10 months maybe But it is very different here the ways things get done here. They couldn't be more different than I am used to. So working, with a big company with remote teams. Some are in India, some are here, some are across the US and Canada Um it just um takes a long time to get things done some times. It takes a long time to get a consensus, track people down etc. so you know I think you know I am not sure I know the answer to that question. I am still trying to figure it out.
P3	Role as a generalist	I've been trying to be a generalist in that field as well so.
	Gorilla tactic – this is the user's explanation of their practice process	It is a very brut force approach to getting the UCD inside the process. What happens is that you know obviously with a lot of this industry, I mean this field, you realize that you have to some [na] sometimes. You have to take it to a level that that's the only way that it's [the UCD] going to get in.

P11	Hallway testing technique	Our hallway usability would be something of that nature. We do some of that where it's you know where we need time to interview actual people and see what they think. Right, because we'll get a lot of you know executives that say this needs to be targeting you know ah soccer moms as an example. Right? We have no soccer moms to talk to.
P5	Hard core developers – roles or a culture in the team?	But I think software, you know, development in general there is some fairly deeply entrenched sort of cultures and ideas. Not like in the hard core software world
P6	High fidelity prototyping within this team	We have used them in the past when the client asks for them. Where they specifically want to see a hi-fi type layout screen mock ups. We will definitely do that. It's not what we recommend. We recommend putting the effort in area like development and ah in planning of ah other areas.
P9	The participant really liked this in their process	I had to say a bottom line I love XP by the way.
P4	The participant would like this in a tool	Maybe there is a way to do attachments or what ever but you know these things tend be like 10 megabits sometime. You know very large files. Like, but having that connection to our day to day stuff and then sort a have it generate a report that says ok here is the end of the 2 weeks, here's the list of all the stuff that people did and here's, here's all the .pdfs right there all in 1 spot
P7	Working independent of the team	In my job now you have a lot more time and you also work more independently.
P5	Interesting process perspective	But you know adopting process to fit the need instead if trying to fit the need into the process um I'll advocate for the former.
P9	Participant working one iteration (at least) ahead of development	I was working 1 iteration ahead of the developers and as for implementing they would say ok this is how I'm doing it, am I doing it right
P12	Participant's iterations are typically this long	Basically 1 cycle we were using 2 week iterations
P12	Iterative explained by Agile participant	So we go in a circular motion. The idea is what are the user goals, How can we capitalize on that and is it technically feasible?
P4	Iterative explained by	We'll first iterate with them within our own team, within our, our design team like with our RAs and other folks

	UCD participant	just to make sure we are all on the same page. We'll often come up with you know maybe 3 different versions of it.
P4	Keeping your eye on the big picture	think about the vision, think about the vision so then we can sort of do that a little bit at a time and, and we're not completely designing small user experience without thinking about the big picture.
P9	Learning what the contribution of UCD to a project is	I'm not saying I'm not saying that there isn't organizational resistance, but it isn't developers. Very quickly developers start to see oh my God this person makes my job easier. Right?
P13	Location	Yes, we were all located in the same building.
P6	Lo-Fidelity prototyping	And we would use Visio, and I became a Visio expert and could do any number of things with that tool, to put together prototypes and process flows and process mapping.
P6	Low fidelity prototyping for the customer's understanding	But after the first couple of iterations you have more functionality that the user can sit down and play with in every session. I involve more people in the observation process cause they can help distill information and bring it back to the business. Typically what would be after a focus group session where we are observing users, is we'll bring all of the information we've gained back from that, present it back to the business. Basically evaluate what's important to the business, to change or improve or you know perhaps leave the same. We present all that back to them and then they determine what they want to do with that information.
P6	Medium-fidelity prototyping	Moving these around to come up with an overall design of what you think a certain screen would look like. And then you take that away and you can um I don't know um what you'd call them maybe medium-fi prototyping where you actually use a tool
P1	Meeting with the customer – face to face	So a lot of the work is not just the design that's really interfacing you know, it's really interfacing with the customer a lot.
P1	Participant is meeting with the customer for UCD reasons	What's very important to do is to meet the customer and really understand the (UCD) problem before ah developing something for them
P7	Evolving a participant's methodology	So a bit of both. The basic structure (their methodology) is there but we do need to evolve because we are a service organization. Right?
P12	Team morale	[Researcher] - OK, and what you're saying basically is then the entire process was basically really positive? So everybody walked away happier.

		[Participant] -oh absolutely, absolutely
P5	Niche strategy – UCD person brought in for a specific fix	But then you know there is a lot of interaction design UI design people work specifically on you know strategic stuff. So this, this application has this usability problem here that you have to work on and that's, that's strategic in a sense that is very tactical and not, not certainly end to end type of project.
P5	Working on projects that are not really Agile	Yeah that certainly as been my experience with you know the times I have been involved with agile type methodology. It is usually sold as that, or a client has requested it. So, the client says, well all right, your team does agile methods, so you guys have to adhere to that. And the people that are selling the project go yeah no problem we all know all about that.
P2	Methodology/process evolution	We had a fairly strong development team but for the most part was open to trying something new. Yeah we got the actual development team kind of excited about this idea. We got people kind a pulled in. We were running a little study group on agile and different people on the team coming and presenting on different aspects of it to the rest of the team. And we brought in some um experts to help advise us and stuff like that.
P10	Use of personas	We use the time to construct the personas and then the developers came in. I had the personas and we decided in the first planning day which person described the best of our target user.
P12	Planning meetings	So if they were looking at doing contextual inquiries type things ahead of time then they would usually be there as a to present the items that were going to be estimated and planned t put in the backlog.
P9	Interacting with the project manager	I would work with the product manager to make sure those things were in iterations. Future iterations.
P4	Problems arise in the project when…	Right, so you know cause in the past you know one Information Architect was on this one project for six months, if he left the company we were screwed right? All of his knowledge went with him
P1	Participant relating to the overall project vision	And there's also just the whole project plan. Like we want to have these features implemented by this date, you know but that's done more by the official project manager per say. And there's like this project plan that will be some very specific, you know, screen by screen details.
P4	Release times	To do it incrementally, to do it in small chunks and not take 12 months to do a major release. You, you roll out a

		new feature here and a new feature there.
P8	Working requirements/features are set this way	so there was you know every 2 week there would be an iteration planning meeting where people would decide the work for the next couple of week
P12	Resolving challenges with customers	So it's a matter of cost and time. So it you know we've noticed we're bringing it down to a certain level where they want it. If you can do it in this time frame we want it.
P12	Resolving challenges with the development team	And I don't think we had anybody with that view point on our team after seeing the contributions of what that UCD group brought to us.
P9	Resolving challenges with the UCD folks	So with the product manager I would because he wasn't a usability person so he was even sensitive to many sorts of usability issues. So there was an education process
P13	Resource costs	Yes, there was an amount of time set aside for us to do our thing [research, contextual inquiry…] that did cost [company name] some money.
P4	Roles	I mean our team is like 8 people and then there is a team of 5 other people that focus on doing the user research.
P1	Scenarios produced for the developers	And sometimes they come up with scenarios that you know may never happen. So sometimes it is [evaporating] around the scenario. And this is when I write test clips about when they do implement a certain feature of the application. Following the test clips is like following the scenario of how somebody might actually use the system.
P4	Scenarios for user/customers	Yeah we will often do a little scenario like pretend that like you are a user here and your goal is to find this one piece of content. So you first come to the first page you try out this link to that page to the next page. Then you pull down a ….you choose from this pull down and you choose another thing and then you know by the 4th screen you have gotten to where you want. So we will often have a little scenario that people can follow thru as apposed, … you know to that, that matches one of our use cases.
P1	Participant's interaction with the client/customer frequency	Well you know it's all kinds of things. Like if they have a regular weekly meeting. There might be times in the project where I would like to go every week. But um you know getting together face to face if at all possible ah then we follow up with a phone or a chat or email.
P8	These processes are separated in the development	And what happened today is the 2 processes [design and development] are completely disconnected, something that is thrown over the wall firm 1 process into the other

	process	process Development team has to it the ground running not waste any time.
P12	What happens at start up	you know we were sort of deriving everything from the beginning
P12	Tandem design and development	Well we didn't end up at the beginning but ended up trying to work more ahead of the development team. Basically 1 cycle we were using 2 week iterations so we would try to stay 2 weeks ahead of the dev team.
P9	Team concerns/issues	So it was sort of like… and then I mean I picked up a little design project and I made a design and I brought it …[development said]oh you completely focusing on the wrong things and this is a really nice design but we don't have time to do this anyway blah, blah. So there were 2 or 3 different ways I tried to interact with them. Proposing my ideas, or asking them for more information on how they were approaching it. On both occasions they were quite closed.
P9	Team membership/inclusion	I think it's really it depends on how you are introduced into the team.
P5	Team size	Let's see how many people do we have in our group? 1 2 3 4 5 let's see 7 8 9. Yeah there are ten of us in this department.
P2	Testing difficulty	That's not a problem if it's a functional test where it's all contained within a kind a single application that's usually ok. But where we are actually trying to get into this situation where we are spanning applications which … and when you get into enterprise systems that's clearly is not uncommon
P1	This technique made the customer happy	Because you've shown your dedication to getting it right by spending time there with them [the customer].
P2	Tool wise I would like this	And as I mentioned we're working with someone to build something like that. We've spoken with [person's name] about this too, on the testing side. There are certain types of tests that we have difficulty in performing. So unit tests aren't a problem. Having automated unit tests, certain types of functional tests we have difficulty in automating, as a single tests that…and if we had a tool ah that could do that, that would be of value to us so…
P6	The tools I use are	Um typically we do low-fi prototyping in an agile project with sticky notes on a white board.
P6	UCD Agile mixed team	In agile dev but also incorporating UCD in a lot of what we are doing in agile projects.
P9	UCDS's concerns	I guess developers have things to do with the software

	about development mindset	that aren't necessarily in a user casing, because you know one of the developers used to say 50% of the software is invisible to the user right?
P12	UCD and development war	So from a methodology stand point it seems like the UCD side of the war is evolving kind of a little bit behind the development side of the world. From a methodology stand point.
P12	UCD has opposition towards development	I would say that there I think there would be more opposition from the UCD group towards the developers then there are from the developers towards the UCD group.
P12	UCD methodology evolution	Yeah um I think that I guess looking at the evolution of usability not just UCD I guess but UCD and interaction design and everything around that general area, is still is it seems to me and this is as an outside observer, and I've never done this job before but it feels to me the field is evolving.
P1	UCD component is in the development process	And I do practice UCD practices with all these things [projects] I've done.
P7	UCD requirements gathering	In the next phase we will flush out any of the tasks that we need to. So we will go back and do more task analysis and detailed GUI design.
P1	UCD requirements gathering issue/communication overcome this way	Even then you might get something solved and then you try it out, you make some rough representation of a UI, you show it to them and then they'll say oh yes you know but it might organized this like this. This would come before that. It doesn't make sense how you have it now.
P11	UCD technique used by the participant	Which gets split up onto 3 areas. So we carry the strategy piece, we do the wire framing piece and we do the … actually 4, we do the visual design and we do the client development. All the way up to you know cutting the HTML
P1	UCD testing step	Well when I first laid out the system you know I have an area of the screen for this type of text, these kinds of navigation buttons for things like that. Just to make sure that overall grid is respected.
P10	Up front stage	So first when the projects didn't start I had 1 and a half months time to inquire, be of knowledge about users and ah build up the personas.
P5	Us and them	There is sort of an "us and them" type of mentality that seems to exist. You know you do get agilest who, who have worked successfully with usability people or design

		people. But frequently you get that sort oh we know what we are doing and you know you get these, these usability people coming in and they slow us down, right?
P5	The user advocate	Yeah definitely, because there are a bunch of competing stakeholders. You know you have business stakeholders. You've got product managers stakeholders. You've got development stakeholders, people who are actually building it. And then nobody in those realms, although they are important, none of them are sort of in the unique position of advocating for the people that are actually gong to be using the product.
P2	Agile adds value to the project	Much of the value behind any of the agile processes I think is the realization that, that change is inevitable and it's not just the result of poor planning.
P2	This is added business value to the project	Right? And sometimes we're doing things purely for business value that the user might not even , you know that is directly seen For example if you imagine that we decide we need to collect certain types of metrics in the application.
P2	Added value	We could get a lot of value if we had something that could really allow us to implement that kind of thing as a single test.
P12	Added value for the user	When they are using the application right. It benefits the user.
P3	Waterfall mentality	That's, that's waterfall thinking to separate those two concerns.
P3	We are this as a team	We were all user centric in the way we thought and worked with them.
P4	We are using this methodology	They're also working agile, so we, we kind have this Agile [UCD] design team working in conjunction with an agile development team.
P4	We found improvement in our process through	It forces us to not think in terms of let's do this let's spend 4 months designing something and see if it's any good. In, in the web world today you cannot afford to do that.
P2	Trying different methodologies	And we've tried a couple of different approaches with this and I don't know if we have a right answer or not..
P5	What team members think about this process developers/UCD	Well I think, you know, I think that the perception it [UCD] that it slows down the process a bit.
P5	Who works on what -roles	Yeah it depends on the company and you know I mean more junior people tend to work on more strategic enhancements. You know tactical stuff where as the experienced people tend to do the end-to-end stuff.
P7	Work experience	Ok. So um I have been in the, I guess the software

		industry, if you want to call it that for about 11 years. I have worked in oil and gas internet IP, medical software in military. I have an undergraduate degree from here, at the University of Calgary in computer science. And no specialization though. I did focus my degree on software um engineering and HCI.

Appendix C – Coded Categories

The following table represents the initial Open codes and how they were categorized during Axial coding. Please note that some of the codes appear in more than one category as they have some relevance to both categories.

DERIVED CATEGORY	CODES
Upfront predevelopment stage and resources allocation	• What helps with the big picture (overall vision) • Contextual inquiry – research specific • Contextual inquiry • Documentation • Keeping your eye on the big picture • Participant relating to the overall project vision • Resource costs • What happens at start up • Up front stage
Roles – who is doing what	• Customer role • Customer voicing requirements • Dependence work – dependencies or non-dependencies in a team environment • End user – their role/ description • Role as a generalist • Gorilla tactic – this is the user's explanation of their practice process • Hard core developers – roles or a culture in the team? • Working independent of the team • Roles • We are this as a team • Who works on what –roles • Work experience
Compromises – who is giving up what	• Participant concerned about building the right product • Compromising methodologies • Consequences of evolving methodology
Passing designs around – between customer/users, development, and UCD folks	• Customer giving requirements to designer • Design is passed from the UCD team to the developers • Design goes to business • Design goes to the UCD team • The design or application goes to the customer or acting customer for approval
Tools –what are used and what would like to be used in the process	• High fidelity prototyping within this team • The participant would like this in a tool • Lo-Fidelity prototyping • Medium-fidelity prototyping • Use of personas • Planning meetings

	• Scenarios produced for the developers • Scenarios for user/customers • Tool wise I would like this • The tools I use are
Team dynamics	• Concerned about UCD process from an Agile perspective • UCDS's concerns about development mindset • Communication issue. • Company size • Dependence work – dependencies or non-dependencies in a team environment • Developers being defensive to other team members • UCD and development war • UCD has opposition towards development • Learning what the contribution of UCD to a project is • Team morale • Resolving challenges with customers • Resolving challenges with the development team • Resolving challenges with the UCD folks • Team concerns/issues • Team membership/inclusion • UCD Agile mixed team • Us and them • What team members think about this process - developers/UCD
Customer/User/Developers/UCD folks/Team communication	• UCD communicated with the developers this often • Communication throughout with the user • Communication with the developers and UCD • Communication between the developers and UCD/bridge UCD • Communication issue. • Communication issue between developers and UCD people • Communicating with the product manager • Meeting with the customer – face to face • Resolving challenges with customers • Participant is meeting with the customer for UCD reasons

	• Interacting with the project manager • Participant's interaction with the client/customer frequency
Testing	• Agile testing perspective the developers used. • Developers testing step happens this way • Hallway testing technique • Testing difficulty • UCD testing step
Methodology Process and Adaptations	• Methodology/process evolution • Tandem design and development • Addressing change and changing the process of development • Adopting a methodology • An Agile model component being used • Agile term - Author specific • Consequences of evolving methodology • Consequences using Agile incorrectly • Doing things differently than in the past and reflecting on it • Doing a methodology or practice without knowing • End-to-end strategy – having a UCD person involved throughout the entire project • Gaps understanding their process • Gorilla tactic – this is the user's explanation of their practice process • Participant working one iteration (at least) ahead of development • Participant's iterations are typically this long • Iterative explained by Agile participant • Iterative explained by UCD participant • Evolving a participant's methodology • Niche strategy – UCD person brought in for a specific fix • Planning meetings • Release times • Team size • UCD Agile mixed team • UCD methodology evolution • UCD component is in the development process • UCD technique used by the participant • We are using this methodology • We found improvement in our process

	through • Trying different methodologies
Requirements Process	• Agile requirements gathering • Agile requirements gathering – overcoming it this way • Customer giving requirements to designer • Customer voicing requirements • Working requirements/features are set this way • UCD requirements gathering • UCD requirements gathering issue/communication overcome this way
Project Process/Methodology results	• Found the project/methodology successful as a result of this practice/variable • Found the project/methodology unsuccessful as a result of this practice/variable • The team member found this to be a positive aspect of the methodology they are using • The participant really liked this in their process Team morale • Problems arise in the project when • This technique made the customer happy • Agile adds value to the project • This is added business value to the project • Added value • Added value for the user • We found improvement in our process through
Miscellaneous	• Individuals process comparison to an Agile author's • This Agile term is used interchangeably with UCD • No available information about a product to be developed • Cart before the horse – putting product technology before the user • Design do's and don'ts • Interesting process perspective • Location • Low fidelity prototyping for the customer's understanding • Working on projects that are not really Agile • These processes are separated in the development process

	• Team size • The user advocate • Waterfall mentality

Table 5: Open codes assigned to their initial categories.

Appendix D – Co-Author Permissions