

# Agile Methods and User-Centered Design: How These Two Methodologies Are Being Successfully Integrated In Industry

David Fox  
University of Calgary  
bdfox@ucalgary.ca

Jonathan Sillito  
University of Calgary  
sillito@cpsc.ucalgary.ca

Frank Maurer  
University of Calgary  
maurer@cpsc.ucalgary.ca

## Abstract

*A core principle of Agile development is to satisfy the customer by providing valuable software on an early and continuous basis. For a software application to be valuable it should have a user interface that is usable. Recently there has been some evidence that suggests using Agile methods alone does not ensure that an applications UI is usable. As a result, there is currently interest in combining Agile methods with user-centered design (UCD) practices. To support existing empirical evidence that these methodologies co-exist effectively we have conducted a study with participants that have previously combined these two methodologies. Our findings, combined with existing work show that the existing model used for Agile UCD integration can be broadened into a more common model. In this paper we describe three different approaches taken by our participants to achieve this integration. We term these approaches the Generalist, Specialist, and the Hybrid approach.*

## 1. Introduction

The Agile Manifesto asserts that “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software” [1]. For a software application to be valuable it should have a usable user interface (UI). However, some evidence suggests that building software aimed at satisfying the customer’s wants does not always produce a usable product [5]. This gives rise to the important research question of how best to produce usable software in an Agile context

Some prior research has addressed the integration of software engineering practices with human computer interaction (HCI) practices. More specifically, the integration of with Agile methods with user-centered design practices (UCD) [2, 5, 7, 8]. One of the

problems surrounding the integration of these two methodologies is that they traditionally take different approaches in terms of how upfront design or requirements gathering resources are allocated.

Agile methods strive to deliver small feature sets of working software into the hands of the customer as quickly as possible in short iterations. At the beginning of each iteration an onsite customer representative’s requests are captured in descriptive stories and written on index cards. According to Ambler, a user story is a very high-level definition of a requirement, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it [12]. Once stories are captured, they are prioritized in order to ensure that the number of stories selected can be completed in the time allocated for that iteration. The stories with higher priority are selected for implementation in that iteration. The priority is based on the business value the customer feels is most important for that iteration. The remaining stories go into a backlog for implementation in later iterations. During the iteration the development team splits the work between the developers and the selected stories are implemented. On completion, the finished iteration is demonstrated to the customer for approval. This process is repeated until the customer determines that additional feature costs are no longer justified in terms of business value [2]. In this way, much of the upfront effort required by many traditional methodologies is significantly reduced or eliminated [3].

UCD, on the other hand, champions a considerable amount of upfront research and analysis prior to development. This research and analysis is based on three kinds of design activities. 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 kind of activities involve 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 kind of activity involves, “iterative design that fixes the problems found by the users in usability testing as part of the product development life cycle.” [4] These design activities are initially carried out before any development is underway. Once development is underway, the activities described above are reused to evaluate the product with users with the goal of improving the product throughout the development life cycle [4].

Although these two methodologies have very different approaches to requirements gathering and the amount of upfront design advocated, they do have some similarities. Both methodologies are iterative in nature. Agile methods build working software for delivery to the customer in an iterative manner anywhere from a couple of weeks to a couple of months [5]. Usability testing refers to observing typical users performing typical tasks in order to measure performance of task completion on a UI [6]. In UCD iterative design is used to correct problems found by users in usability testing with low fidelity prototypes [4]. Both methodologies are also human-centric development approaches. As the name suggests, UCD is aimed at developing software with the user in mind. Agile methods involves an onsite customer representative to create a feedback loop with the development team. This is aimed at building software that customers want.

To contribute to an understanding of how these practices can be effectively combined, we have conducted an interview study with members of teams that have integrated user-centered design practices into an Agile methods development process. We have analyzed the different approaches taken by our participants to integrate these approaches into their design and development process. We have also compared our findings with previous work [1, 5, 7, 8] to describe a general UCD Agile practices integration model. Our analysis also describes three different approaches, the Generalist, Specialist, and the Hybrid approach to better understand the roles the various team member have. We also confirm the previous model proposed by Sy is in fact being used by development teams in industry.

The rest of this paper is organized as follows. In the next section we look at recent work that has been done in our area of research for later comparison with our findings. Next we discuss the research methodology used in our study (see Section 3). This is followed by the results of our study (see Section 4). Finally we conclude the paper with a brief discussion of those results (see Section 5).

## 2. Related Work

Some recent research examines UCD and Agile methods to determine if and how these two approaches can coexist in the development process. We briefly discuss this work and will later compare their results with our findings.

Patton [5] discusses the motivation for adapting UCD practices into his agile development process. The development team was using eXtreme Programming (XP) practices. The team included expert end-users that were product managers working for the same company. They were building software at an aggressive rate with “deliveries that were on time and the expected scope intact.” However, they also found that the resulting product had features that the end-user did not want and was missing features that the end-user did want. To correct this problem, Patton describes the ten-step process that he and his team added to their current development process to produce an Agile Usage Centered Design process.

His ten-step process identified participants that were included in the design process. These included domain experts, business people, programmers, and test/QA staff. 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.

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 that it did leave his team with valuable tacit knowledge. What his paper does not cover in any detail is the actual implementation process after the design has been completed. In other words, he does not describe how the design was passed from the design stage to the implementation stage [5]. We pursue this topic in more detail in our findings.

Meszaros and Aston describe in their practitioner’s report the process they use in adding usability testing into an agile project [7]. Like Patton’s approach, Meszaros and Aston did not include a UCDS on their team. Instead, they had two developers on their team acting as the UCDS. Their approach included typical UCD practices such as low fidelity prototyping and usability testing to facilitate a final UI design. This process meant that the developers were also the UCDS. That is, they followed a generalist approach to design where a developer is assumed to have enough UCD expertise to take on the UCDS role.

Meszaros stated that adding UCD practices was of value to his development process. In fact he claims,

“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”.

His paper discusses the implementation of their process over a short period of time. What his paper does not discuss is how their process effects development over the long term [7].

Ferreira, Nobel, and Biddle investigate four different projects in relation to Agile methods iterative development and UI design [8]. Their approach is qualitative in nature using Grounded Theory, which we describe in detail in our Method section. Using this method they derive certain themes that emerge from their data. These themes include were iteration planning affects UI design, iterations drive usability testing, usability testing results in changes to the development, and finally Agile changes the relationship between UI designers and software developers.

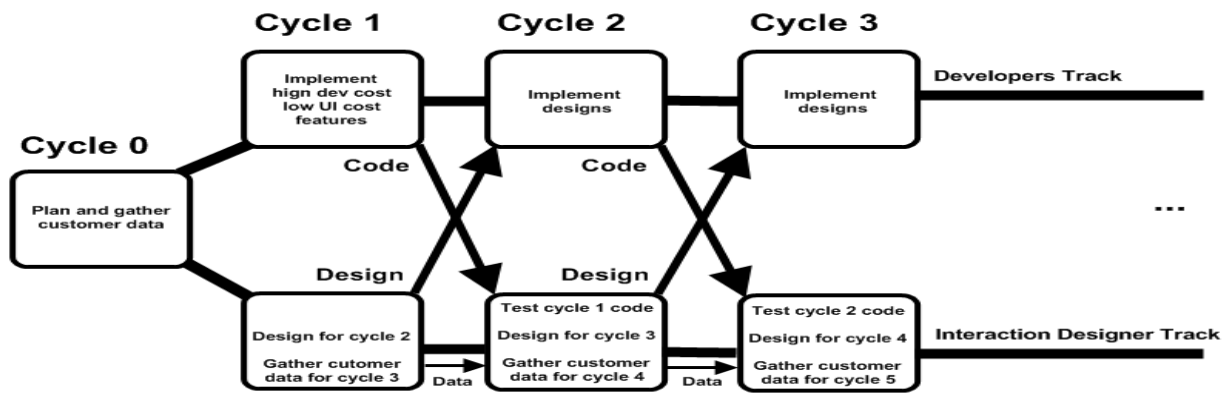
Their paper also briefly describes the roles of the team members in the four projects. In projects one, two, and four, the teams consisted of developers and UCDS, case three had a developer acting as the UCDS. In the three cases that employed a UCDS, the UCDS interacted with the customers to derive the 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 [8].

The paper very briefly discusses the processes that the different teams used. More detail may have given

more insight into a generalized approach of all four projects integrating these two processes.

Finally, Sy [9] describes the process of integrating UCD with agile methods currently being successfully adopted by Autodesk. In their process, Sy refers to iterations on design as “cycles”. Cycle zero is used to acquire initial information as about the project by conducting a contextual inquiry. 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 the 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 [9]. Typical activities include contextual interviews, observation, reconstruction of previous events or tasks performed, and discussions with the users [6].

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 previous performed contextual inquiry. An initial design is conceived and sent for implementation by the developers in cycle one.



**Fig. 1. Sy’s interaction designer/developer tracks diagram**

Once in cycle one, the UCDS designs prototypes and conducts usability testing to refine 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. “*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 [9].

Sy’s article provides a detailed view of the process that is used at Autodesk. The process is very much in keeping with some 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 wanted to investigate a general approach of methodology integration based on multiple companies to determine if it is successful in general.

The above work has significant information of how teams are integrating Agile methods and UCD practices. In our study we wanted to look at both detailed implementation approaches as well as over multiple teams to determine a generalized approach as well as the different approaches teams are taking.

### 3. Research Method

This paper reports on a qualitative study using grounded theory approach [10]. The study involved members of teams integrating agile methodologies with UCD practices. We conducted in-depth, semi-structured interviews (either face to face or by telephone) with ten participants from Canada, the United States, and Europe. The participants had varied backgrounds and played different roles on their teams. All interviews were performed by the first author of this paper. Interviews lasted between 36 minutes and 64 minutes. An audio recording of each interview was made and each recording was transcribed verbatim.

We conducted 10 interviews and we refer to the participants as P1...P10. Participants P1, P5, P7, P9, and P10 are UCD specialists, each of whom had formal UCD training. Participants P3 and P8 had some informal UCD training. Participant P6 is a business analyst with some exposure to Agile methods and informal UCD training. Participant P2 is a chief innovation officer with both formal UCD training and agile development experience. Finally, participant P4 is an information architect with formal UCD training, technical development skills and agile methods experience. All of the participants were on different

development teams working for different companies with the exception of P3 and P6 who worked for the same company but in different countries and on different projects. It is important to note that the majority of our data was collected from UCD specialists with a smaller amount of data being collected from agile developers and that data set of 10 is smaller in size from other grounded theory research. Another aspect of our research that may be a limitation to the study is that we interviewed one person on each of the ten teams. Perhaps interviewing one developer and one UCD specialist per team may have generated a more complete view of the process.

All of the data we collected was qualitative and to structure our data collection and analysis we have used a grounded theory approach [10]. The grounded theory approach consists of iterative data collection and by analysis with the goal of producing a theory to explain a situation of interest. Our analysis involved various coding activities.

We first performed open coding. This coding consists of attaching specific code words, developed during the open coding process, to portions of the data. The goal of open coding is to identify important concepts in the data and categorize the data based on those concepts. We performed open coding on our interview transcripts using HyperRESEARCH, a qualitative research-coding tool. Examples of codes we developed include: *user advocate*, *niche strategy*, and *UCD Agile mixed term*.

The next stage, axial coding, assembles the previously attached codes from the open coding stage into core relationships to each other. 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 gathering further data for analysis, which then becomes the final core categories. While performing axial coding we derived a number of preliminary core categories. Examples of core categories include: upfront predevelopment stage, team member communication, and passing UCD design around.

The final stage in grounded theory is selective coding. 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. In this way we derived the overall concepts or themes that emerged from the core categories. The findings section discusses these concepts and themes in terms of how the participants employed Agile and UCD.

## 4. Findings

The analysis of our data produced a number of interesting insights into the approaches that the participants took when integrating elements of agile methods and UCD practices together. It is important to note that none of the processes described by our participants are identical. However, the approaches they used have some commonalities. We used these commonalities to construct an overall picture that describes the common aspects of the approaches taken by the participants.

### 4.1 Commonalities

First we discuss the commonalities discovered in the data was that the participant's processes had an upfront stage for UI design that occurred before any software development began. In this stage, which we called the Initial Stage, we found two main activities, contextual inquiry and low fidelity prototyping. One participant described these two activities as the "discovery stage followed by a prototyping stage" [P1]. These two activities are illustrated as Iteration 0 in Fig 2. The initial stage had "*some measure of user research*" [P3] involved. This was carried out by a UCDS or a team member acting as a UCDS. The UCDS performed contextual inquiry to better understand who the users were and their needs in terms of what tasks they needed to perform.

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 roughly 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. 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.

As mentioned in the introduction section, one of the key differences that Agile methods and UCD have is the allocation of upfront resources for UI design. While traditionally UCD design would have aimed for a nearly complete set of features or requirements before development begins [11], we found in our study that the Initial Stages were time boxed for short periods of time. This meant there was less time for the initial analysis and testing. As a result, this shortened the upfront UCD design process and, hence, a small set of

features was derived for the first development iteration. One participant remarked that their Initial Stage lasted two weeks [P8], while another stated theirs had lasted only a few days [P3]. However, the data analysis showed that for most participants the Initial Stage lasted approximately four weeks. The output from the Initial Stage was complete an 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 iterations One through Three in Fig 2. 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. 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”.

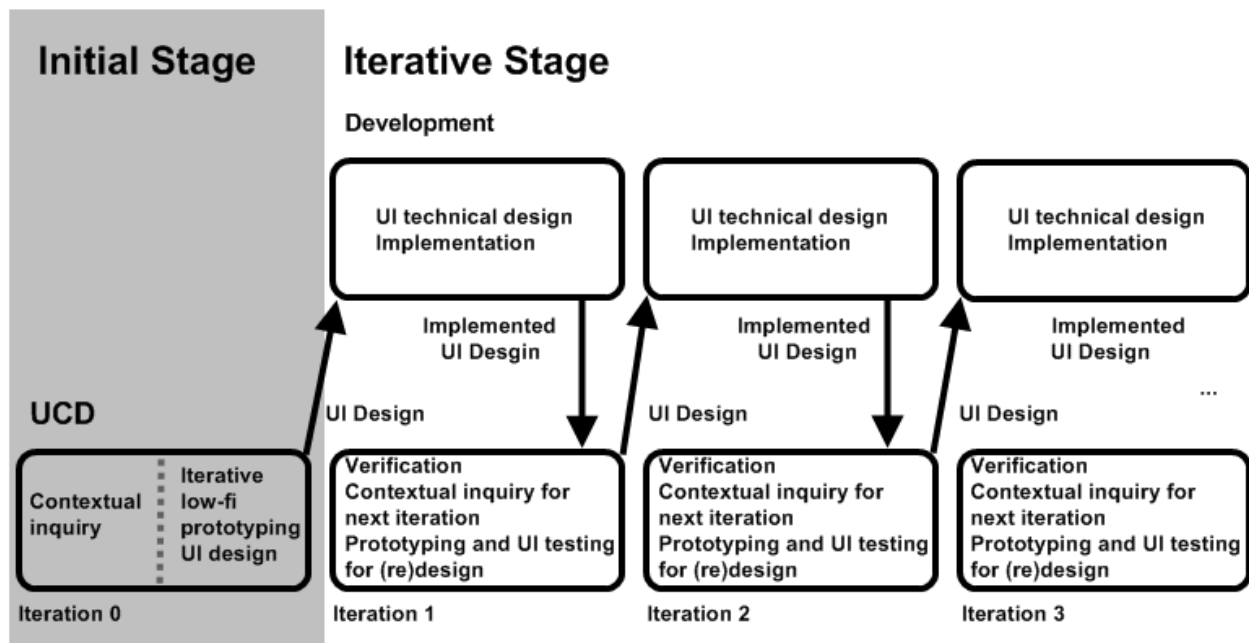
Usability testing, which typically followed verification, may or may not include user participation. One participant remarked that when users were not available for testing the team members did a collaborative UI inspection in order to determine if the user's tasks were possible to accomplish in an effective manner [P3]. Participants P2, P3, P5, P6, P8, P9, and P10 all claimed they used actual end-users to verify and test implemented features.

If the implemented features are verified as correct and they 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 2). This process continues iteratively for the duration of the projects lifecycle.

The general approach discussed above, discusses similarities of approaches taken by development teams and the UCD specialists. These similarities represented the interaction between the two during 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. Although these approaches were similar, there were differences that are worth noting. The differences lie in who executes certain aspects of the process, i.e. who performs which roles in the process.



**Fig. 2.** A UCD Agile methods project development life cycle common to the majority of 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 that continues for the remainder of the project lifecycle.

Fig. 2, closely resembles Sy's diagram. Both show the initial upfront stage of UI design as well as the iterative

parallel designer and developer timeline. Our study was inconclusive in terms of when design was passed back

and forth in terms of an exact timeline. In other words, UCD and developers designs were both passed during and at the end of iteration, but this was not the same for every participant. We therefore show a more linear diagram to represent the general diagram.

## 4.2 The Specialist Approach

We call the first approach the *Specialist* approach. It consists of three main member groups: the users customers<sup>1</sup>, 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 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 step begins. This step involves producing primitive representations of a UI’s features and testing those low fidelity drawings and/or wireframes with users to determine features for implementation. Low fidelity prototyping tools ranged from pen and paper, sticky notes, white boards and 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. 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 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 four weeks whereas P9 stated this typically lasted six weeks. P1, P9, and P10 also remarked that once the Iterative Stage begins, the UCD effort was shortened to two weeks as opposed to up to six weeks in the Initial Stage.

With the development team implements the features for the iteration, the UCDS then 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”. Concurrently, usability testing occurs during this time to augment, extend and refine the feature list for the next development iteration. P10 remarked that 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. For example P10 worked on two different projects. On one project the developers were collocated with him and the exchange of design was constant and ongoing, sometimes daily. On the other project that he work on the developers were not in the same city. In that case the design was passed at the end of each iteration. The UCDS then takes that implementation back to the customer or user to perform usability tests. This testing differs from usability testing of low-fi prototypes in that those tests were testing for future features, whereas the current tests evaluate completely implemented features. If the implemented feature 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 not follow the above approach. P7 typically worked on waterfall projects that were military based. He stated that because the projects he worked on were funded by the military the “red tape” required him to gather requirements in a waterfall-like process. He did however follow the *Specialist* approach in that he was the bridge between the developers, users, and customer. He also was knowledgeable of Agile methods and expressed the need for an Agile process in order to improve their development process.

## 4.3 The Generalist Approach

The *Generalist* approach uses only two main roles, the users/customers and the developers acting also as UCD specialists. It is worth noting that the developers were not formally trained UCDS. However, they did have some informal or self-taught UCD expertise. 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

---

<sup>1</sup> Customers refer to anyone that has a vested interest in the development of the project. Users refer to people that are actual end-users of an application.

members acting as UCD performers we found on a team was two [P8]. The other teams had multiple UCDS. 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, P8]. In 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 case of P3's and P6's team, all developers contributed to the UCD activities. 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 complete. 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 completed 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 Generalist and the Specialist approach is the roles that the developers need to practice. The data suggested that the working environment in the Generalist's approach was much less formal than that of the environment of the

Specialists. This may have been due to the way the UCDS was introduced in the Specialist approach.

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 specialist 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. Although this poses some very interesting questions, we leave further research and discussion of this team membership issue for later work.

#### **4.4 The Generalist Specialist Approach**

P2 and P4 followed a slight deviation to the Generalist approach. Their team had a group member which had both formal UCD training and software development experience. 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 Specialist Generalist's team had more than one UCD specialists, which were managed by the Specialist Generalist person. In the Specialist approach there was only one UCD person on the team with limited or no development skills. Thus for the purposes of this paper we will call the team member with both formal UCD training and software development experience a hybrid team member.

This approach was very similar to the Specialist and Generalist approach in that it followed the same practices mentioned in 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 between all the different team members including the developers, the UCD team, and customers. For example, P2 stated that the UCD folks 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.

It is important to mention here that both P2 and P4 worked for very large international companies, both have extensive experience in their fields. The size of the company is important because of the politics that need 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 features that would be implemented in the next iteration. For this process facilitation process, the team was not present and P4 acted as a bridge for delivering information back to the developers and UCDS. The main difference that this approach had was that the hybrid member was not actually acting as the developer or the UCD specialist directly but mitigated between the two groups that never directly communicated.

## 5 Discussion

Our study set out to investigate how agile methods are currently being integrated with UCD practices as well as validate Sy's previous work. We first started by examining related work. We found that although these studies provided valuable insight for our study, there were some areas that needed to be further investigated. A broader empirical basis is needed to make more general conclusions.

With exception of Ferreira's work, all the previous studies centered around one team or company. However, there was no strong evidence that these approaches were general enough for use in different companies or teams. Ferrier's project studies, on the other hand, did span 4 projects and teams. However, the process descriptions were not detailed in terms of how the UI designs were passed around from group to group in the development process. Our study examined both of the above to find commonalities all the teams shared for integrating agile methods with UCD. All teams followed similar processes. This confirms the model originally presented by Sy [9] and suggests that this integration approach is widely applicable. In fact three of the participants were employed at multinational companies that are using this model on multiple teams. Our participants' teams, however, did have some differences in terms of team roles and responsibilities. We also presented different approaches regarding who performed the UCD-

oriented activities in a team: the Specialist, the Generalist, and the Specialist-Generalist approaches.

Sy's approach had UCDS members on their team that gathered some upfront data when a new project was started much like in the Specialist approach we presented. Once cycle zero was completed, the developers and UCDS worked in parallel during the cycles that followed [9]. This approach closely matched Specialists approach of this study.

In Patton's team, there was no formal UCDS. The development team was expected to contribute to the UI design. Therefore, the development team was responsible for both UI design and development [5]. His approach was very similar our Generalist approach.

In Ferreira's 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, resembled the Specialist approach with at least one UCDS. Project three resembled the Generalist Specialist approach in that the team member acting as the UCDS was also a developer. There was no clear indication that the UI designer was formally trained in UCD practices. She only states that UI design was their interest. All of the related work discussed above did fit into one of our approaches. Moreover, all of the related work shows similarities to the process model found in our study data. 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 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 that there was value added to their development process.

Finally, as mentioned in the introduction, a key difference between these Agile methods and UCD is the upfront resource allocation for planning and developing UIs. Our findings show that these differences were overcome in all the approaches in the same way. Some upfront effort was invested (the Initial Stage) by all participants. But this upfront design effort was rather limited compared to traditional UCD processes. On average, our participants spent approximately four weeks on the Initial Stage.

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.

We also saw that all of the approaches did have at least one UCDS or a team member acting as a UCDS. Obviously, expertise on UCD is beneficial when user centered design activities are performed. The teams of all our study participants performed some research, paper prototyping, and usability testing prior to any feature implementation. So it seems that in order for these agile and UCD methodologies to be integrated both must compromise their upfront resource allocation.

## 6 Future Work

Our study provided insight to what was being done when integrating Agile and UCD practices and broadened the empirical basis of research in this area. Besides presenting three different role structures in our paper, we reconfirm and replicate results from previous studies. Our research also left us with some open research questions that may be worthwhile pursuing at a later date.

For instance, the Generalist approach couples the developer's development duties with UCD practices. Does this mean that the Generalist approach is more efficient because of the elimination of that team member and a reduced need to document the result of UCD activities for the development team? In other words does removing the middleman between the developer and customer expedite the Agile UCD process? A drawback of the generalist approach might be that developers taking the role of UCDS are not deeply trained in UCD. The question posed is; what do developers need to know about UCD to be effective in this role?

Another open question is are the requirements or feature gathering elicited from the customer more effective in terms of user's wants and needs because they were done so specifically by a UCDS? Or can a Generalist without formal UCD training elicit requirements and features with the same level of effective results as a UCDS?

We found that all but two participants felt that adding UCD to their software development process was valuable. An interesting question is exactly how the participants found the addition valuable?

And finally is the Generalist Specialist more effective at designing UIs because they can bridge both the UCD design perspective and the technical development perspective?

1. "Principles behind the Agile Manifesto" [Online] 2001 Available: <http://agilemanifesto.org/principles.html> [Accessed: April 2, 2008]
2. F. Maurer, P. McInerney, "UCD in Agile Process: Dream Team or Odd Couple?" in *Interactions*, vol. 12, no. 6, 2005, pp. 19-23.
3. W. Royce "Managing the Development of Large Software Systems." in *IEEE WESCON 1970*, pp. 328-338.
4. C. Barnum, "What is usability and what is usability testing" in *Usability Testing and Research*, Longman, New York, NY USA, (2002) pp. 7-30.
5. J. Patton, "Hitting the Target: Adding Interaction Design to Agile Software Development" in *Proceedings of OOPSLA 2002*, pp. 1-7.
6. J. Preece, Y. Rogers, H. Sharp, "User-Centered Approaches to Interaction design" in *Interaction Design: Beyond Human-Computer Interaction*. John Wiley and Sons NY, 2002, pp. 296-323.
7. G. Meszaros, J. Aston, "Adding Usability to an Agile Project." in *Proceedings of Agile 2006*, pp. 289-294
8. J. Ferreira, J. Nobel, R. Biddle, "Agile Development Iterations and UI Design." in *Proceedings of Agile (2007)* pp. 50-58
9. D. Sy, "Adapting Usability Investigations for Agile User-centered Design". vol. 2, no.3 *Journal of Usability Studies* May 2007 pp. 112-130 2007
10. A. Strauss, J. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. London, England: Sage Publications 1998 pp. 12, 101-143
11. R. Baecker, J. Grudin W. Buxton, S. Greenberg, "How To Design Usable Systems" in *Human-Computer Interaction: Toward the Year 2000*. San Francisco CA: Morgan Kaufmann Publishing, 1995 pp. 93-120
12. S. Ambler "User Stories" [Online] Available: <http://www.agilemodeling.com/artifacts/userStory.htm> [Accessed: March 21, 2008]