Towards a Framework for Integrating Agile Development and User-Centred Design

Stephanie Chamberlain[1], Helen Sharp[2], and Neil Maiden[3]

[1] Bit10 Ltd, Sovereign Court, Sir William Lyons Road, Coventry CV4 7EZ, UK
stephanie.chamberlain@gmail.com
[2] Centre for Research in Computing The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK
h.c.sharp@open.ac.uk
[3] Centre for HCI Design City University, Northampton Square London EC1V 0HB, UK
n.a.m.maiden@city.ac.uk

Abstract. Due to a number of similarities between user-centred design (UCD) and agile development, coupled with an appreciation that developers are rarely usability experts, it seems attractive to integrate these two approaches. However, although agile methods share some of the same aims as UCD, there are also distinct differences. These differences have made the use of these methods on development projects problematic. This paper reports a field study designed to investigate the use of agile methods alongside UCD in one particular organization. The aim of the study was to develop a framework for use by project teams wishing to integrate UCD practices with agile development. The study, its findings and five principles for integrating UCD and agile development arising from this work are discussed.

1 Introduction

The importance of knowing who the users are, understanding their priorities and goals, and actively involving them in uncovering requirements (e.g. [10]) is well understood in software engineering. However the role they should play, how they should be involved, and how much they should be involved has been a matter of dispute (e.g. [6, 9]). User involvement is also a central concern of HCI, and the importance of integrating software engineering and HCI methods has been recognised for many years (IFIP WG 2.7/13.4). The Agile Manifesto emphasises the importance of involving the customer in a development project, but this practice is proving to be problematic (e.g. [12]), and it is rare for a real end-user to take the role of customer.

"User Centred Design" (UCD) is an approach which aims to involve the users in a meaningful and appropriate way throughout a system's development (e.g. [5], [15]). Gould et al [5] first proposed three principles of UCD in the mid-1980s, and in the 20 years since then, various techniques for involving users successfully have been developed. Integrating UCD and agile development therefore has the potential to help agile developers with the difficult practice of involving customers, and the wider concern of how to integrate HCI concerns with software engineering.

The purpose of the study described in this paper was to identify and investigate the issues faced by a project team trying to integrate UCD and agile development. The study reported was conducted within one organisation where Scrum, XP and UCD were being used. We report the study and its findings, and extract five principles which appear to be significant for successfully integrating UCD and agile methods. In the rest of this section, we explore UCD, other approaches to integrating agile methods with UCD principles, and compare UCD and agile approaches. In the next section we describe the method, and in section 3 we present our results. Section 5 presents the five principles, and the paper concludes with some practical suggestions.

1.1 User-Centred Design (UCD)

The term UCD refers to both a collection of techniques and the philosophy at the heart of these techniques. The overall philosophy of UCD is to place the user at the centre of the design process through the use of rigorous methods. For instance, the designer tries to “get to know” the users initially through techniques such as interviews, direct observation in context, forums and questionnaires, before moving on to design prototypes for the users to test within a real-life context. Often the first “prototype” is simply a paper one which the designer constructs through an analysis of the tasks that the user will perform. As development progresses and more sophisticated prototypes are developed, the user may be asked to perform tasks using the prototype with only minimum guidance from the tester. The results are then fed into an iterative process which continues until a final version of the system emerges.

1.2 Integrating UCD and Agile Development

The potential of XP to provide a bridge between software engineering and HCI is not a new idea. A discussion between Kent Beck and Alan Cooper [13] concluded that there were indeed strengths of Interaction Design and XP that could be combined. Beck and Andres [3] acknowledge this by including an interaction designer in the agile development team; personas are now commonly used in agile projects (e.g. [1]).

Several other approaches to integrating HCI and agile concerns have been suggested. For example, Kane [8] proposed how ‘discount usability’ [14] may be integrated with agile development. Ambler [1] suggests several models which can be used to facilitate interaction between users and developers and shows how these can be used in an agile project. Holtzblatt et al [7] have proposed a modified version of contextual design (rapid contextual design) which is appropriate for projects with a shorter timescale, including agile development [4].

1.3 Similarities and Differences Between UCD and Agile Development

A project involving both Agile Methods and UCD becomes a challenge because although there are several similarities, there are also distinct differences (e.g. [17]). The three main similarities are:

1. They rely on an iterative development process, building on empirical information from previous cycles or rounds. For instance, one of XP’s values is feedback ([2:20]), and the idea of refactoring code is an embodiment of this value. In UCD one of its founding principles is iterative design.

2. Agile techniques place an emphasis on the user, encouraging participation throughout the development process. For instance, in Scrum, user evaluation of the product is encouraged on a monthly basis as users are ideally present during the sprint review ([16:54]) and the "Product Owner" is responsible for the requirements and feature prioritisation for the product. A second founding principle of UCD, is early and continual focus on users.
3. Both approaches emphasise the importance of team coherence. Beck states that one of the purposes of the planning game is to "bring the team together" ([2:85]). One of the features of the UCD approach is that the whole team should have the user in mind while developing the product.

The two main differences are:

1. UCD advocates maintain that certain design products are required to support communication with developers, while agile methods seek minimal documentation.
2. UCD encourages the team to understand their users as much as possible before the product build begins, whereas agile methods are largely against an up-front period of investigation at the expense of writing code.

2 Fieldwork

2.1 Method

Three project teams in one organisation were observed for around 2-4 hours per week on site by one individual for a period of 6 months. The organisation hosting these projects was a large media company with a tradition of employing a user-centred approach to development. The organisation had a clear distinction between 'designers', who were responsible for user-centred activities, and 'developers' who produced the code. The observer was a member of staff at the organisation, but not a member of any of the project teams that formed the basis of the study.

The study period was divided into two parts. During the first part, which lasted about a month, the researcher identified some themes which appeared to be significant to the projects being observed. These themes were then used as a framework for a more in-depth investigation which took up the remainder of the observation period.

The initial approach to observation was ethnographic in nature in that the researcher approached the activity as 'strange' and had no *a priori* hypotheses to test. The initial themes emerged over the first period of study. The observation strategy combined shadowing of individuals with site or situation observations such as meetings (14 observation sessions in all). Ten interviews were carried out in order to gain further insight into the observations and therefore not all of the team members were interviewed, although care was taken to gain as much of a cross-section as possible across all teams. Most regular meetings for all three teams were attended and some unannounced visits were made in order to gain a deeper insight into the day to day workings of the teams. At the start of each meeting or observation the method was briefly explained. The team knew they could cease the observation at any point and that the observer would leave without need of an explanation.

Contemporaneous notes, photographs and some video recordings were used to record the interviews and observations. After each session, a summary of key points was written. The environment, interactions and process were recorded by the observer.

Documents helped to provide evidence that the processes which had not been observed but were reported through interviews, e.g. maintaining a sprint backlog graph, were actually being carried out and documented.

2.2 The Project Teams

Three projects formed the main focus of the field study work. Here we refer to them as Project I, Project S and Project M. Each team contained developers (coders) and designers (those who traditionally worked on the user research and usability). Table 1 summarises the projects and their approach to integrating agile and UCD.

Table 1. A summary of the three projects observed through our study

Features	Project I	Project S	Project M
Project Application	Website to involve people in local civic life, including online community to promote and re-engage a political audience.	Interactive TV application: a two-video stream interactive quiz designed to complement a TV programme.	Web-based message board facility for the study organisation.
Methodology followed	UCD and Scrum	UCD and XP	UCD and Scrum
Main User Group	Members of the public	Members of the public	Members of the public
Other Users[1]	Content editors for the website	Administrators/ Editors	Administrators/ moderators[2] The "product owner"[3]
Distribution of the project team	All on the same floor	Spread over two floors	Seated together

All three teams had experience of using agile methods with UCD in the past, and had developed their own approaches to integration, which were observed in this study. These are described below.

The designers of Project I had reported problems on previous agile projects where they had used Scrum. They believed that these stemmed from the inclusion of the design team in the Scrum from the outset of the project. They also felt that they needed an "upfront" period of user research. On previous projects, few usability recommendations had been implemented and the team felt they had been lead by technical requirements over and above user and business requirements. Consequently, Project I decided to change their approach so that the designers did not enter the Scrum until there was clear value in doing so. The team envisaged:

[1] Observations showed that users within the organisation were often seen to be user representatives on all three projects.

[2] The employees within the organisation that supported the message-boards by ensuring no illegal or inflammatory content appeared.

[3] This person's job was to ensure the requirements for both the moderators and the end-users were fulfilled.

- A separate “up front” period of user requirements gathering and research which took place before development began.
- A prototyping stream where the developers and designers worked together.
- A three-man design team where one designer fed the Scrum with prototypes while the other two designers carried out user research.
- The use of iterative usability testing with constant feedback throughout the development phase.

In Project M, the designers had found attending the Scrum with developers on past projects to be unhelpful and so they ran their own UCD process in parallel to the developers’ use of Scrum.

Project S were part of the Interactive Television Department where they were required to deliver within very tight timescales due to fixed transmission dates. They had found that XP worked best for them and the team were using this approach when the observations took place. During the study the team admitted that some UCD tools and methods are occasionally overlooked as a result of external time pressures.

The three projects therefore had different approaches to using UCD with an agile approach. Project I attempted to integrate UCD and Scrum, Project M used UCD and Scrum in combination and tried to align the processes, and in Project S the designers used UCD and activity progressed quite separately from XP development.

3 Results

Four themes emerged from the initial observation period: user involvement, collaboration and culture, prototyping and the project lifecycle. These appeared to be significant issues faced by the project teams in working within UCD and agile development. The meaning of these themes, and the results of further investigations focused around these themes are presented below.

3.1 User Involvement

Through our observations, user involvement was characterised as being where:

- the users were invited to give opinions or test prototypes
- the users were interviewed, observed or questioned for research purposes
- the user’s interaction with the product was considered in detail

Each team used different tactics for ensuring that they had suitable user involvement. Project I developed personas based on earlier user research, and then developed a user journey, i.e. usage scenario, for each persona. They also analysed usage patterns taken from the existing version of the website. This gave them an idea as to how far the users were getting through certain processes such as setting up a campaign. Usability issues were raised in meetings by editorial staff.

In Project M we only saw one user testing the system during the observation period. As Project M involved the development of an internal system for managing web message boards, the user in this case was an editor within the organisation. Interestingly a member of the team said that the testing was being carried out “for the developer”. The Editor was testing a part of the system to ensure it fulfilled her team's requirements before it was released. The Product Owner was observed attending the sprint review and following this, there was a demo of the work carried out in the sprint. This was mainly for the benefit of the Product Owner who asked questions and the developers proudly showed off the work they had done. This was done at the desk and in an informal way rather than through a formal presentation. It was one of the Product Owner's responsibilities to prioritise features within the sprint. An example "Sprint Backlog" on a whiteboard is shown in Fig 1 from Project M.

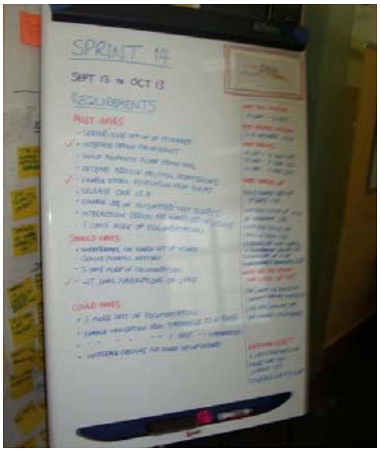

Fig. 1. Sprint Backlog

Project S showed the least evidence of user involvement. However, the user's interaction with the product was seen to be important and the user's needs were often represented by user representatives taken from the team. For instance, the broadcast assistant was observed playing the role of the customer in order to carry out what appeared to be user acceptance testing before the product went to the dedicated QA team. The functional specification was said to be made up of a variety of "user experiences". Stories were written out on cards against the functional specification as the development producer explained what happened at each stage in user terms. The specification was written from the user's perspective.

3.2 Collaboration and Culture

Collaboration was observed with relation to:

- The collaboration between individuals within the team
- Specifically, the collaboration between designers and developers
- The culture that the chosen methodology created

Project I held cross-functional meetings which included representatives from the development, design and editorial teams. The team worked collaboratively in the meetings and requirements were captured from all team members. There had been problems with collaboration between developers and designers in the past; in this project, the Design Lead commented that "we need to get everyone involved in the user journeys as this was the problem before". There was evidence of a struggle for power between the two groups, as shown by this exchange recorded in our notes:

The Scrum Master claims that a developer has already done the back-end work. The Design Lead asks incredulously "based on what spec?" One of the developers replies that it was based on the spec provided by the technical lead. It was agreed that a general meeting was required amongst the leads of the project over this particular issue.

Each group seems to be guarding themselves against having to deal with decisions being made by one group at the expense of another. However, later this defensiveness is displayed again by the Scrum Master who objects to the Product Owner asking probing questions about estimating during a Sprint Review.

The Product Owner refers to the graph and says that the shape of the graph seems to indicate that generally there is an under-estimate of how much work there is to be done. The rather defensive retort from the Scrum Master is that the estimates are not inaccurate but that the requirements change. He adds that if there hadn't been so many small tasks to complete on top of the list of tasks for the sprint, then the team would have delivered all of the tasks by the end of the Sprint. The Product Owner states that there is always new work that crops up therefore it might make sense to say that a certain % of time is allocated for these changes. The development producer adds that there are two options: Either we need to accept that this happens and plan for it or stop it happening if it stops people working effectively. The Scrum Master again defensively says that they already are working effectively.

On Project M there were similar issues between the designers and developers. However, in this project the design team split away from Scrum altogether - this was only used by the developers. The developers sometimes pair-programmed in order to solve hard problems but this was ad hoc, not regular. Many of the problems encountered with collaboration were not to do with the use of Scrum or UCD on this project but largely due to other factors such as lack of people resources.

On Project S communication between the designer and developers was mostly informal. Meetings involving the designer and developers together were scarce. Team meetings often did not involve the designer because they were arranged at the same time as other meetings she had to go to. As a result, there was a disconnection observed between the designer and the developers.

3.3 Prototyping

Each of the projects used prototyping; Project M used an evolutionary approach, Project S used a throw-away approach, and Project I used a combination of both evolutionary and throw-away prototyping.

Project I faced timescale pressures which left little time to handle prototyping effectively. For example, the client-side developer noted that there was not much time for reviewing things as "priorities on the project have been set elsewhere". The cycle of prototyping and feedback didn't work in the way that had been envisaged.

Project M also faced time problems with prototyping, but caused by the different timescales associated with paper prototyping versus development prototyping. This meant that the designers had a shorter iteration cycle than the developers. Ultimately this may have contributed to the abandonment of Scrum by Project M designers.

The Usability Engineer observed that "design prototyping is faster than development prototyping as the <development> languages we use are too slow to prototype in. You ask for a prototype and 6 weeks later you get it." The designers worked at a different pace to the developers which made it hard to iterate around versions of software or designs.

On Project S, the broadcast assistant and a developer used a paper prototype to test that the application supported users' tasks as expected. This prototype was a series of storyboards and flows.

3.4 Project Lifecycle

The different projects exhibited different ways of combining the traditional lifecycle phases with the agile approach. For instance, Project M dedicated a whole sprint to requirements gathering at the start of the project. Some development was planned during this time but the phase was named a "business analysis phase" to indicate that the emphasis was on requirements gathering.

On Project I, the designers advocated "up-front design methods" where significant user research is carried out before any coding is done. The designer's tasks at the start of the project were to: analyse usage patterns, create user journeys (including personas), map the user's mental model and create a high level specification. Based on this information they then prioritised the task list and sent it to the rest of the team.

This activity itself wasn't observed in our study although the resulting artefacts and their use were in evidence (see Figure 2). The lead interaction designer was keen to get the whole team involved in the user journeys as not doing this had caused problems on previous projects. Each area such as technical, editorial and design had a "Discipline Lead" who looked after the interests of that particular group within the project. Requirements gathering was carried out by all of the discipline leads together.

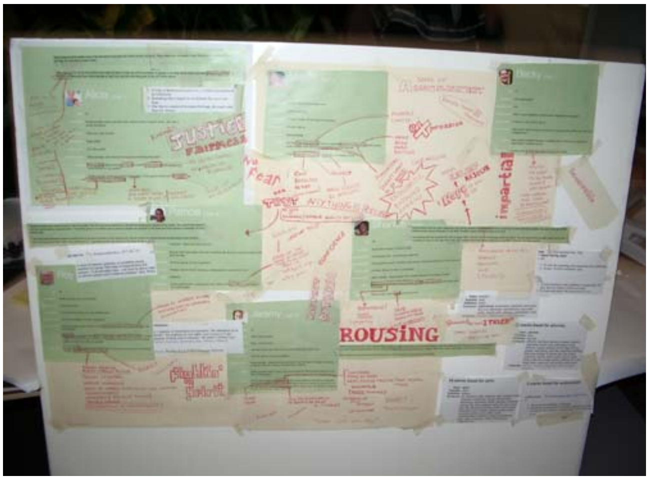

Fig. 2. Results Board (from user research in Project I) showing user opinions

In Project S, an application functional specification was produced before the "planing game" and this provided the basis of discussion. User journeys had also been produced at this stage. Project S were unhappy making decisions on the customer's behalf.

For instance, during an observed multi-disciplinary team meeting, a developer suggests that they should plan a story around the 'red button' (which navigates to interactive TV from linear TV channels) but others are unwilling to do so until requirements had been gained from the customer.

In Project M, a whole sprint was given over to requirements gathering. Some development was planned during this time but the phase was named a "business analysis phase" to indicate that the emphasis was on requirements gathering.

4 Discussion

All projects had some degree of design before coding started but the one most loyal to XP (Project S) had the shortest design period. It also had the least user interaction. However, it must be noted that this project had a much shorter timescale than the others observed.

Project I seemed to have least problems with collaboration which may have been related to the fact that UCD and agile principles were well integrated.

Project M suffered from a detachment of the development and design methodology which as a result tended to operate separately. There was a culture of defensiveness which may have grown up out of this segregation of the two disciplines.

In reviewing the three projects it seems that a fundamental problem of communication exists between the developers and designers within each team and the subject of power within the project is a tricky one. Designers within a project defend their discipline in response to decisions made by the developers, and vice versa.

The power aspects of UCD and Agile are interesting as part of the reason these methodologies came about was because each discipline needed a defence mechanism against other disciplines such as management, or the business taking away their power. Consequently, some kind of balance needs to be put in place to ensure that this power struggle is controlled on a project.

Prototyping also appears to be problematic due to the timescales involved in developing an application in comparison to the design of a paper prototype. However, this may be ameliorated if the designers were managed differently so that other projects were interspersed for the designers and there didn't seem to be so much lag between feedback and implementation.

5 Five Principles for Integrating UCD and Agile Development

Based on our observations and the themes discovered, we have evolved a set of five principles which are significant where UCD and agile methods are to be integrated:

1. **User Involvement** – the user should be involved in the development process but also supported by a number of other roles within the team, such as having a proxy user on the team.
2. **Collaboration and Culture** – the designers and developers must be willing to communicate and work together extremely closely, on a day to day basis. Likewise the customer should also be an active member of the team not just a passive bystander.
3. **Prototyping** – the designers must be willing to "feed the developers" with prototypes and user feedback on a cycle that works for everyone involved.
4. **Project Lifecycle** – UCD practitioners must be given ample time in order to discover the basic needs of their users before any code gets released into the shared coding environment.
5. **Project Management** – Finally, the agile/UCD integration must exist within a cohesive project management framework that facilitates without being overly bureaucratic or prescriptive.

Although these principles have arisen through the observation of one particular organisation attempting to integrate agile methods with UCD, they can go some way to offer other teams a framework with which to begin. More research is required in this area through the observation of further organisations and project teams.

6 Conclusion

User-centred-design and agile methods are compatible, and they can work together but they can also provide problems if the key principles aren't addressed. For instance, the two methodologies can be at odds due to:

- Power struggles between developers and designers
- Time differences between designers' and developers' capacity to create tangible outcomes from each iteration round. Development usually takes more time
- Communication issues if members of the team don't take part in some elements/phase of the project
- A reluctance to understand the needs of each element of the project
- The extent to which the user is able/willing to contribute to the project

However, these can be overcome if:

- There is some balancing role or mechanism put in place to ensure that each discipline has equal power on the team
- Resource management and project management ensures the management of time and resources equate to utilised resources that don't become frustrated whilst waiting for results
- All members of the project team are available/involved at each key point of the project
- The user plays a part in the project so that their requirements are catered for and that the end-product works in a realistic situation

If agile methods and UCD are successfully integrated within a project team, the evidence from our observations suggest that it will be more likely to deliver benefits to the business and most importantly to the user as well.

References

1. Ambler, S. (2002) *Agile Modeling*, John Wiley and Sons
2. Beck. K. (2000) *Extreme Programming Explained_* United States and Canada, Addison Wesley.
3. Beck, K. and Andres C. (2005) *eXtreme Programming Explained: embrace change* (2nd edition), Addison-Wesley
4. Beyer, H. Holtzblatt, K. & Baker, L. (2004) An Agile Customer-Centered Method: Rapid Contextual Design, in *Proceedings of XP/AU 2004,* eds C. Zannier et al, LNCS 3134 Springer-Verlag.
5. Gould, JD and Lewis, CH (1985) Designing for Usability: key principles and what designers think, *Communications of the ACM*, 28(3), 300-311.

6. Heinbokel, T., Sonnentag, S., Frese, M., Stolte, W. & Brodbeck, F. C. (1996) Don't underestimate the problems of user centredness in software development projects - there are many!, *Behaviour & Information Technology*, 15(4), 226-236.
7. Hotlzblatt, K., Wendell, J.B. and Wood, S (2005) *Rapid Contextual Design: A How-to Guide to Key Techniques for User-Centered Design*, Morgan Kauffman.
8. Kane, D. (2003) Finding a place for discount usability engineering in agile development, *ADC 2003*, pp40-46.
9. Keil, M. & Carmel, E. (1995) Customer-Developer Links in Software Development, *Communications of the ACM*, 38, (5), 33-44.
10. Kotonya, G. & Sommerville, I. (1998) *Requirements Engineering: processes and techniques*, John Wiley & Sons.
11. Kujala, S. (2003) User involvement: a review of the benefits and challenges, *Behaviour & Information Technology*, 22(1) 1-16.
12. Martin, A., Biddle, R., and Noble, J. (2004) The XP Customer Role in Practice: Three Studies, in *Proceedings of ADC 2004*, Salt Lake City, June.
13. Nelson. E. (2002) [Internet] *Extreme Programming vs. Interaction Design* http:// www.fawcette.com/interviews/becknelson_cooper/ [Accessed September 2004]
14. Nielsen, J. (1993) *Usability Engineering*, Morgan Kaufman.
15. Preece. J., H.Sharp and Y.Rogers (2002) *Interaction Design: Beyond Human Computer Interaction* New Jersey, John Wiley & Sons. Inc.
16. Schwaber. K. and M.Beedle (2002) *Agile Software development with Scrum* New Jersey, Prentice Hall.
17. Sharp, H.C., Robinson, H.M. and Segal, J.A. (2004) "eXtreme Programming and User-Centred Design: friend or foe?" in *HCI2004 Design for Life*, Vol 2.