# Up-Front Interaction Design in Agile Development

Jennifer Ferreira[1], James Noble[1], and Robert Biddle[2]

[1] Victoria University of Wellington, New Zealand
{jennifer,kjx}@mcs.vuw.ac.nz
[2] Human-Oriented Technology Laboratory
Carleton University, Ottawa, Canada
robert_biddle@carleton.ca

**Abstract.** In this paper we address how interaction design and agile development work together, with a focus on the issue of interaction design being done "up-front", before software development begins. Our study method used interviews with interaction designers and software developers on several agile teams. We used the qualitative approach of grounded theory to code and interpret the results. Our interpretation includes appreciation for benefits seen for a certain amount of up-front interaction design, and benefits from some levels of interaction design continuing with the iterations of software development.

## 1 Introduction

Interaction design and agile development have much in common, most importantly the fact that they will often both be involved in development of the same software. Despite this, there has been little investigation or discussion on how the two processes work together, and the issues that arise. We have been conducting studies of software teams that use both interaction design and agile development in order to better understand practice. In this paper, we focus on the issue of interaction design being done "up-front", before software development begins.

Interaction design and agile development have different perspectives on software. Whereas interaction design has a focus on how the end users will work with the software, agile development has a focus on how the software should be constructed. Both have major roles in making good software. The two processes also have in common an appreciation for the importance of evaluation of customer satisfaction, and how an iterative approach is the best way to accomplish this. However, how these two iterative processes are combined is unclear.

In the next section we outline other literature that addresses this issue. We then present our study method, team profiles, and our results, categorizing and quoting findings from our interviews. We then integrate these findings and produce an initial interpretation of the practice that emerges from our studies. We then present our conclusions and plans for future work.

G. Concas et al. (Eds.): XP 2007, LNCS 4536, pp. 9–16, 2007.

## 2 Background

The way in which interaction designers[1] and agile developers should work together has been discussed surprisingly little. The debate between Kent Beck and Alan Cooper [1] did explicitly address the issue of when interaction design should occur relative to software development. They agree on many things, but Cooper argues that all the interaction design should be done before any programming, and Beck disagrees.

> Cooper: "So when I talk about organizational change, I'm not talking about having a more robust communication between two constituencies who are not addressing the appropriate problem. I'm talking about incorporating a new constituency that focuses exclusively on the behavioral issues. And the behavioral issues need to be addressed before construction begins."
>
> Beck: "The interaction designer becomes a bottleneck, because all the decision-making comes to this one central point. This creates a hierarchical communication structure, and my philosophy is more on the complex-system side — that software development shouldn't be composed of phases. ... The process, however, seems to be avoiding a problem that we've worked very hard to eliminate. The engineering practices of extreme programming are precisely there to eliminate that imbalance, to create an engineering team that can spin as fast as the interaction team."

Jeff Patton describes in several papers and tutorials how interaction design and agile development can work together by using a process where interaction design iterations fit in the iterative structure of agile development [2]. Lynn Miller describes similar experience in managing projects where the interaction design was of critical importance to the software [3]. Her approach is that both interaction design and programming use a common process, where the two kinds of work are done in parallel, but are one iteration out of phase. In this way, the interaction designers are doing detailed design for the iteration that the programmers will do next, and doing evaluation of the iteration that the programmers did last. Chamberlain, Sharp and Maiden [4] use a field study to ground their introduction of a broad framework for how interaction design and agile development can work together. In particular, their study shows, and their framework explains, how the general values and practices typical in interaction design and in agile development are quite similar and can assist teams in working together, but that efforts must be made to ensure balance, appropriate resource management, participation, and in general a coherence of purpose.

[1] Our interviewees used the terms interaction designer, user interface designer and UI designer variously; we use the term interaction designer to refer to the member of the development team, whose main responsibility it is to design the user experience and the user interface. The team members involved in mainly coding activities are referred to as the developers.

## 3 Method and Participants

Our research method was qualitative, using grounded theory based on interviews. We conducted our study using semi-structured in-depth one-on-one interviews with team members from software teams at several different companies, each in different countries. Our aim was to study actual practice, rather than any ideal or experimental situation. We selected teams that we felt confident would be regarded as using an agile process, and where the project did involve interaction designers. For each team, we interviewed both someone who concentrated on user interaction design and someone who concentrated on programming. The interviews were voice recorded and transcribed in detail. All persons interviewed were asked to validate the transcriptions. We began our analysis with the method known as open coding, and is used to identify the different categories present in the text. We then performed axial coding, where the relationships between the categories are established, and then began to build up the structure of what we found in the interviews.

The first team, T1, is based in the United States, and develops and markets web-based software for IT professionals. T1 is an XP team consisting of ten engineers and one product manager/user interface designer. At the time of the interviews, T1 was working on redesigning and enhancing one of its products. Their product manager/user interface designer described the previous version of the project as being "hacked together" and having user interaction that was "very cumbersome", and the next version was to address these concerns. We interviewed the engineering manager and product manager/user interface designer.

The second team, T2, is based in Ireland. They develop and sell software to support wealth management. T2 is also an XP team and includes four engineers, one domain expert/on-site customer and two interaction designers. One of the interaction designers explained that there were several smaller projects, but that their main effort was on a kind of application described as a "wealth planner", where both the size and the impending release to their first customer were their concerns at the time of the interviews. We interviewed their project manager and an interaction designer.

The third team, T3, is based in New Zealand and develop software that controls fruit sorting machines. Their team consists of five developers. Their main project was described as not only controlling machinery and reading sensor data, but "gathering this all together and presenting that information to our customer." We interviewed two developers, one of whom had a background in interaction design.

The final two participants, P1 and P2, are both employed by the same software consulting company based in Finland. At the time of the interviews, P1 was an interaction designer working on a system to manage teaching and course scheduling; the team consisted of a project manager, four developers and two interaction designers. P2 was a team lead/developer on a team consisting of five developers, who worked across several projects. P2's main project was developing a new web-based application.

## 4 Results

In this section we present some of the main concepts that emerged in our interviews that relate to the issue of up-front interaction design. In each of the subsections below, we identify a significant pattern, and provide some relevant passages from the interviews. We provide a more general interpretation in the next major section.

**There are Advantages to Up-Front Interaction Design.** The participants were clear about the advantages of doing interaction design before implementation begins. They saw up-front design as having a positive impact on the final product's user satisfaction and consistency, saying it helped mitigate risks and helped designers come up with the best possible design, while keeping to the customer's budget. Up-front design was also seen to contribute to cost and time savings by ensuring better project estimation and prioritization.

"And we have no help lines, or no support lines or anything like that to take calls on how to use the system . . . So that's all because of the designers' up-front work, because of up-front design and also because of the agile process we use." — *interaction designer, T2*

"Just create some up-front consistency, like, what do buttons look like, where are they placed, what do tables look like, how do users interact with tables, what do forms look like, how do you get from a table to a form and then back to the table, like, basic interaction models. So, kinda like a style guide. And I did some of that before I even started." — *product manager/user interface designer, T1*

"The benefits are the fact that doing up-front design, you are not limited by any kind of technology. Of course we take into account the implementation technology. We're not going to do web interfaces that are not possible to do on the web, but we can go on the very, very bleeding edge, so that we can know that we're still in the limits but we're trying the best we can, so it creates, in this way, best designs." — *interaction designer, P1*

"There's also the fact that if you do up-front design, you can take your time for doing the design, meaning that if it would be tied into the iteration, the problem is if you design a part of the system and you don't know what the whole system is, then later on it might happen that you know new requirements of the system, they require a major refactoring of the user interface, something really huge, they might turn the whole concept upside down." — *interaction designer, P1*

**Do Most, Though Not All, Interaction Design Up Front.** Participants from most of the teams believed that having the user interface a certain percentage complete before development begins is essential, but also enough. A user interface that was not completely 100% specified up front left room for decisions to be made during implementation. All participants in the study brought up the fact that there were inevitable changes to the user interface during development, due to implementation issues or issues that were not known up-front.

"Before it gets into development, the user interface is more or less, ninety percent defined." — *interaction designer, T2*

"What we currently try to do here at [Organization], or what I try even sometimes to force through, is that interaction design should be completed at least ninety five percent of the whole system before starting the implementation at all because otherwise it's simply not going to work." — *interaction designer, P1*

"The UI designer actually can get away with not putting all the details and everything into it. Many things just work out during the iteration planning or during development . . . he [the UI designer] doesn't have to make this absolute, final, ultimate thing that is then given to someone. You can get away with a seventy to eighty percent implementation." — *engineering manager, T1*

"[Change is] driven by the needs of the system or new things learned during the project and risks that didn't get identified in the beginning." — *team lead/developer, P2*

**Much of the Interaction Design Involves Study of the Clients and Users.** Participants emphasized the close collaboration that interaction designers had with business oriented people within their own organization, such as marketing, and with the clients and end-users before development begins.

"We'll hold a workshop with the project sponsors, we'd have them with end users of the product, we'd have them with IT people, with the actuaries in the company, with the compliance people, as many as we can who will have input into the product or are using it in some form, or who are developing it in some form. They come from our workshop. We gather the user stories via that." — *interaction designer, T2*

"So we're kind of using the user interface design as the requirement for the developers, because the idea of trying to come up with the user interface design while doing the same piece of software has proven that it is simply impossible. The current deal with the customers is that they . . . we have now budget for this user interface design part and that is kind of labeled as requirements analysis." — *interaction designer, P1*

". . . we use informal conversation with the customer, so someone tries to understand the domain that they're working in, what the problems are and why they're trying to achieve what they're doing." — *developer, T3*

**Interaction Design is Informed by Software Implementation.** Even with some up-front design, interaction designers did gain more insight as the software was actually implemented. In some cases, this allowed fine details to be polished, but in other cases it suggested changes in the higher level design. Also, implementation sometimes revealed that interaction designers would overspecify what the programmers were meant to tackle in an iteration.

"And with XP it's like, these are my cards, this is what I need to design for. I don't care about the next release. I'm not even thinking about that. And nine times out of ten for us, if we did try to think about the next release then when we designed it for that then we didn't end up implementing those features anyway. So I think the fundamental change in thinking is more just for this release . . ." — *product manager/user interface designer, T1*

"With the user interface you gather as much information as you need to do some kind of real thing and then you put it through an iteration ... you know enough about the kinds of interactions you need to perform a particular function so you do that, feed it back into real code and then you've got something that people can play with and look at and then you can go through another iteration or another cycle of 'Is that any good? What should we be thinking about there?' So you've got real working software that people can reflect on a lot better. And you can go through a number of cycles that are purely paper prototyping cycles as well." — *developer, T3*

"... it's not realistic and not a good way of working to try to specify things to the nitty gritty detail, meaning that there will always be some kind of feedback from the developers when they find out that, 'Hey, this is difficult to do,' or 'Have you thought of this kind of a situation, which came up now while trying to implement this?' They give a seed for a need for redesign or completing the design, which is not sensible to do beforehand, because there are so many of these exceptional situations that the user interface designer would never guess, because he would need to know the internals of the system." — *interaction designer, P1*

**Cost and Time are the Issues.** Participants saw no problem with up-front interaction design, as agile development only warns against up-front code design. They saw up-front interaction design as being very different to up-front code design, e.g. up-front code design produces a lot of costly waste, whereas it is essential to be able to refine the interaction design with the customer up-front to make sure it is correct, is essential. Most importantly, there is an understanding that the issues of cost and time are the *reasons* that determine whether the interaction design should have iterations with a prototype rather than programming. Using a light-weight prototype, in the form of pen and paper sketches or PowerPoint slides, was much quicker to develop than an actual application, and still allowed the interaction designer and development team to go through the design together for valuable feedback.

"With faster and quicker iterations in agile, maybe sometimes you need the slower ... With the user interface it takes longer to get feedback, so it doesn't always line up." — *developer, T3*

"The kind of issues that you get in up-front code design are not the issues you get in up-front interaction design at all. Code design comes out of just the amount of waste you get from people doing two, three jobs. Doing the same job two or three times in different forms ... There's an awful lot of waste involved in that, which is the main thing XP was trying to fix in the first place, but with regards to up-front interaction design, you know, putting together screens, in Photoshop or whatever and iteratively running it by customers and things, to make sure that the design itself is correct. So it's a whole different domain, basically." — *project manager, T2*

"We're not writing a specification that is not true, we're trying out the system in the cheapest possible way in a very agile fashion but we're doing it with pen and paper and it's to build a system. This pen and paper thing is not design

up-front, it's defining what the system is in a faster and a cheaper way. If it would be as cheap to implement the system at the same time, well then go for it. But if I can draw in five minutes so much user interface that it takes two months to implement, I really do not think that that is the best way to tackle things." — *interaction designer, P1*

## 5 Interpretation

Working through our interviews, we found a structure emerged. The first step involves the interviewees' conviction about advantages of up-front interaction design. In particular, they held that there are problems mitigated with up-front design, such as poor design judgements that lead to costly redesign or no added value for the customer; budget issues; poor task prioritization; costly redesign problems uncovered late in development, due to new or changing requirements; usability problems; inaccurate work estimates. It was also regarded as understood how these problems are mitigated with up-front design: the team gain a high-level understanding through development of a style guide and navigation model; designs are kept technology free, but without the team completely disregarding the implementation technology. The whole system can be designed very fast, and the team can obtain early customer input about the system.

The second step involves agreement that although much interaction design should be done up front, this is not as simple an idea as it might appear. One important consideration is that what might be called "design" in fact involved close work with business analysts, markets, clients, and end users. This is necessary for several reasons: to determine the value of the business case for development, to appreciate the goals of the client, and to understand the work and mental models of the end users. This raises an important question about our shared terminology. Our participants agreed that this work was part of what they regarded as *design*. On the other hand, this kind of work might also be seen by developers as constituting *analysis* rather than design. Agile development advocates are wary of up-front design because it represents premature commitment, but if it is analysis and does not involve commitment, then it does not constitute the same kind of danger.

The third step involves understanding that there are benefits to interaction design that come from iterative development and delivery of working software. There is acknowledgment that even up-front interaction design must be done in the light of the software platform capability. And beyond this is an understanding that software development provides a different lens through which the interaction design can be examined, helping interaction designers identify strategies for improving and refactoring. Finally, there is also agreement that while prototypes do allow designers to return to clients and end users with something to evaluate, there are advantages to having actual software instead. Not only does it provide more functionality, but also more confidence that value has been delivered.

In summary, there is a conviction of advantages in up-front interaction design. But this turns out to include much that might be regarded as analysis. Moreover, there are also advantages to doing some interaction design together with software development iterations. The real insight is shown in the reasoning about why and when up-front design is advantageous and when it is not. This is shown to involve concern for risk management, not unlike that underlying agile development itself. In essence, up-front design is appropriate if it reduces risk, and inappropriate when it increases risk.

## 6 Conclusions

In this paper we have reported on our studies of up-front interaction design in agile projects. We employed a qualitative research strategy to better understand the practice in actual software development teams committed to both quality interaction design and an agile process for development. We presented some samples of our findings, and our initial interpretation. These reflect not our suggestions for practice, nor any ideal practice, but rather actual practice, together with understanding about how it came to be.

We find that up-front interaction design is commonplace in agile development, and indeed there is agreement that most interaction design be done up front. However, we also find that in interaction design, much business and end-user analysis is understood to be included. Moreover, there is recognition that there are benefits to some interaction design being done in the iterations together with the software development. Most important, there is evidence of understanding that the issue is not whether interaction design should be done up-front or not, but rather when up-front interaction design will reduce risk, and is therefore advisable, rather than increase risk by making premature design commitments.

## References

1. Fawcett, E.: Extreme programming vs. interaction design. FTP Online (2002)
2. Patton, J.: Hitting the target: adding interaction design to agile software development. In: OOPSLA '02: OOPSLA 2002 Practitioners Reports, p. 1. ACM Press, New York, NY, USA (2002)
3. Miller, L.: Case study of customer input for a successful product. In: ADC '05: Proceedings of the Agile Development Conference, pp. 225–234. IEEE Computer Society, Washington, DC, USA (2005)
4. Chamberlain, S., Sharp, H., Maiden, N.A.M.: Towards a framework for integrating agile development and user-centred design. In: XP, vol. 143–153 (2006)