Agile Development Iterations and UI Design

Conference Paper · September 2007

DOI: 10.1109/AGILE.2007.8 · Source: IEEE Xplore

CITATIONS
98

READS
411

3 authors:

Jennifer Ferreira

30 PUBLICATIONS **465** CITATIONS

SEE PROFILE

James Noble

Victoria University of Wellington

393 PUBLICATIONS **5,833** CITATIONS

SEE PROFILE

Robert Biddle

Carleton University

285 PUBLICATIONS **4,392** CITATIONS

SEE PROFILE

Agile Development Iterations and UI Design

Jennifer Ferreira, James Noble
Victoria University of Wellington
Wellington, New Zealand
jennifer@mcs.vuw.ac.nz, kjx@mcs.vuw.ac.nz

Robert Biddle
Carleton University
Ottawa, Canada
robert_biddle@carleton.ca

Abstract

Many agile projects require user interaction (UI) design, but the integration of UI design into agile development is not well understood. This is because both agile development and UI design are iterative — but while agile methods iterate on code with iterations lasting weeks, UI designs typically iterate only on the user interface using low technology prototypes with iterations lasting hours or days. Similarly, both agile development and UI design emphasise testing, but agile development involves automated code testing, while UI must done by expert inspectors or ideally potential end users. We report on a qualitative grounded theory study of real agile projects involving significant UI design. The key results from our study are that agile iterations facilitates usability testing; allows software developers to incorporate results of those tests into subsequent iterations; and crucially, can significantly improve the quality of the relationship between UI designers and software developers.

1. Introduction

Many agile projects require user interaction (UI) design, but the integration of UI design into agile development is not well understood. We have undertaken a study of the actual practice of UI design in agile development, and in this paper we present the aspects of the study that focus on how UI relates to the iterations that are the heartbeat of agile development.

We begin by reviewing the background of this topic, including work specifically on integration of UI design and agile development, but also on the special nature of agile design. We then introduce our study, explaining the qualitative method we used, and outline the scope. In the main section we present the results of the study, identifying the main themes that emerged, and quoting the participants from the interviews we conducted. We then provide our interpretation of these results, and present our conclusions.

2. Background

There has not been any widespread discussion and consensus about the relationship between UI design and agile development. In 2002, however, Fawcett [3] published a debate on this subject between Kent Beck, creator of Extreme Programming (XP), and Alan Cooper, creator of Goal-Directed Design, a well-known approach to UI design. While there was much agreement, the main point of difference was that Cooper advocated UI design being done entirely before any development, using an iterative approach with lightweight prototypes because working software would be too expensive and time-consuming. Beck argued that the two iterative processes might work together and that use of XP practices meant that working software would be cost and time effective in comparison with prototypes.

More recently, Chamberlain, Sharp and Maiden [1] have conducted a field study to explore a framework for understanding how interaction design and agile development might work together. They illustrate the broad issues on which the two processes are similar, most importantly an iterative approach and a strong focus on delivering value. Their study also identifies challenges, including the need for management and resource support for both processes, communication between and participation of both practitioners on a project, and the sharing of their focus on delivering value.

Expert practitioners have also written on their approach to introducing UI design to agile development. Jeff Patton [8] had described how he arranges the interaction design iterations to work within the iterative schedule of agile development. Lynn Miller [7] has documented her experience, where the UI design drove the development process, but where the two processes were interleaved in such a way that UI design iterations addressed the next stage, while development addressed the last stage.

In considering how UI design and agile development might work together, a key issue relates to the idea of "design". The term is generally used to describe the UI pro-

cess, but in agile development even the word is regarded with caution. The agile manifesto [5] includes the preferences "working software over comprehensive documentation" and "responding to change over following a plan", and these are often interpreted as a reduction of emphasis on advance design. Most emphatically, agile development has typically warned of the dangers of "Big Design Up Front" (BDUF), suggesting that the more the design is determined up front, the more difficult it is to change later on, when time and experience may make change desirable.

Consideration of the place of UI design in agile development is therefore affected by the nature of UI design as a kind of design, and the attitude to design within agile development. UI design includes the term "design", and the term has persisted through changes in nomenclature and disciplinary emphasis from user interface design through user interaction design and more recently to "user experience" design. In fact, UI designers, including those in our study, will sometimes refer to their work as simply "design", implicitly meaning the UI design, the aspects of the software that the user will see and with which they will interact. At the same time, software developers will use the term "design" to mean the design of the internal structure of the software, the aspects of the software that affect functionality and execution performance. It is important to realise these differences in nomenclature. In particular, it means that agile software developers who have ideas about the places of design tend to apply them to UI design, even if they have little experience of the UI design process.

In considering the danger of BDUF, it is important to understand that UI design, even when done entirely prior to software development will also be an iterative process. Iteration has always been a characteristic of User-Centered Design, by far the dominant approach to UI design, and a central feature of the codification of approach in ISO standard 13407 "Human centred design processes for interactive systems" [6, 2]. The iterative nature relates to creation and evaluation of the user interface and how it will be used, rather than code, and the evaluation can be done using lightweight prototypes rather than functional software. These prototypes are typically quite low-technology, often sketches done by hand or with simple tools, and the iterations can be very fast: a matter of hours, rather than weeks.

In much of UI design an emphasis is placed on a consistent and coherent design, and the cautious attitude to design in agile development can seem to UI designers either mysterious or simply wrong. Even within the agile development community the nature of design itself has been examined. In particular, Martin Fowler, in his 2000 paper asked "Is Design Dead?" [4]. In considering an answer to this question, he distinguishes "planned design" and "evolutionary design". In particular, he highlights how iterations and refactoring can mean that evolutionary design can turn out well. He explains:

> In its common usage, evolutionary design is a disaster. The design ends up being the aggregation of a bunch of ad-hoc tactical decisions, each of which makes the code harder to alter.

But then goes on to say that the practices of Extreme Programming make a significant difference:

> These enabling practices of continuous integration, testing, and refactoring, provide a new environment that makes evolutionary design plausible.

While this process may seem novel for the design of the internal structure of software, it is not unusual in UI design. While UI design may work from various guidelines and metaphors, the evolution of the design amid iteration and evaluation is an accepted aspect of User-Centered Design.

This background presents some interesting questions about how UI design and agile development might work together. Should one entirely precede the other, or can their iterations be merged or meshed? Does the agile caution about BUDF apply to UI design or are these two different issues? Our study set out to understand common practice.

3. Method

Our research method was qualitative, using grounded theory based on interviews. We conducted semi-structured in-depth one-on-one interviews with team members from software teams at several different companies, each in different countries. For each team, we interviewed both someone who concentrated on user interaction design and someone who concentrated on programming. The primary objective of these interviews was to understand the process and practices relating to user interaction design on software development projects that follow XP.

The interviews were voice recorded and transcribed in detail. All persons interviewed were asked to validate the transcriptions. The interview transcriptions were analyzed first using the method known as open coding. This method is the first step in grounded theory analysis and is used to identify the different categories present in the data. 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.

In the sections below we first describe the overall approach in each organisation. We then present a summary of the main results that relate to the issue of user interaction design, quoting the interviewees as illustration of what emerged, explaining our interpretation.

4. Project Profiles

Our study is based on interviews relating to real projects, and no two real projects are exactly the same. In our study, for example, some of the projects involved completely new software and new UI designs, while others were new versions of existing software and existing UI designs. While the themes we present in the next section did emerge from our interviews, each project had its own story. In this section we present some of these stories to highlight the variety of actual practice.

Project 1

The first project, P1, develops and markets web-based software to support IT managers and development professionals. Based in the United States, this is an XP team consisting of ten engineers and one product manager/user interface designer[1].

P1 involved a new version of existing software. They did not do any overall UI design up front, and their UI designer would have the UI design (HTML mock-up) ready only for the next iteration, sometimes for the next release, designed only for the user stories for that iteration or release. Before the next release (sometimes during the previous release), the user stories and the UI were fleshed out in order to obtain high-level estimates for that release, avoiding these activities from holding up the development process. It was understood that the UI could and should change during development. The development team and UI designer discussed UI issues and their implementation on a daily basis during development, so both the UI designer and the development team understood how the UI design worked and how it was changing. Overall, the developers had significant input into the UI design.

Participants mentioned that the UI could change due to features being cut out and changed during the development process, and so they believed it saved them time to not have a complete UI designed up front. Their UI could also change due to customer feedback or as a result of a usability study. Before an iteration planning meeting, the UI designer and the engineering manager would go through the UI and the user stories it implemented for that iteration. Ideas for the UI design were exchanged at that time. Then the interaction designer would go through the UI and user stories with the whole development team, making sure that the developers understood the UI, and again developer feedback and discussions sometimes led to changes to the UI design. Next, in the iteration planning meeting, developers and the UI designer had further discussions about the UI and if the resulting changes were not too major, they could still be incorporated into the UI design. The UI designer also made sure that the UI design was based on how much work it was going to take to implement that design, and that the developers did not have to spend too much time on UI implementation. At this meeting the UI designer, engineering manager and project manager decided on which cards would go into an iteration. Then the cards were implemented by the pair programmers and once they had finished, the cards were ready for acceptance.

During acceptance testing the team walked through the UI features manually, checking their usability and how well they worked. The team had the HTML mock-up projected on one screen and the actual application projected on another screen side by side. Large fixes were then written as new user stories for another iteration and smaller fixes were given back to the developers to fix straight away. The UI was iteratively designed and implemented with an overall goal or metaphor in mind — an idea that was never written down, but was understood by everyone in the team. This idea guided the development of the UI. It was seen as a waste of time to design too far ahead, as during the development process features were often changed or dropped and the discussions between the UI designer and the rest of the team could lead to changes in the UI.

Project 2

The second project, P2, is based in Ireland. They develop and sell software to support financial management. P2 is also an XP team and includes four engineers, one domain expert/on-site customer and two UI designers.

Once P2's marketing department had verified that the product was viable, then the UI designers started the UI design of the product. XP was seen to begin only after the UI design had been completed, and the UI designers had passed the product on to the developers.

According to P2's participants XP began with the development planning meetings. During the first meeting of the process, cards were created from the user stories and prioritized to determine the order of development. Each card took 2 weeks to implement. In this pre-release planning meeting, they also estimated when the product was going to be released. Development of their products usually lasted three to four iterations in total and each iteration lasted 2 weeks. Before each iteration, an iteration planning meeting was held. P2's participants described their implementation process as a "tight development cycle."

Within the iterations, the developers performed unit testing and refactoring every five minutes and created builds hourly. As a component of the product was implemented, all the tests, including the acceptance tests for that component were created at the same time. The UI designers

[1] The product manager/user interface designer will be referred to as this team's UI designer throughout the rest of the paper.

then tested these builds daily. They checked the cards that were implemented the day before, from the perspective of the UI, and if they were happy with them the builds were signed off. If not, feedback regarding changes or omissions were given to the developers. Every few days the domain expert tested the UI too. In both instances large fixes could result in new cards being created, in which case the domain expert, developers and UI designers would negotiate about which iterations would include the new cards.

UI issues were dealt with in the same way as any other development issue was dealt with: for large fixes new cards were created, new price calculations made and all cards were re-prioritized for iterative implementation. However, since the UI was 90% defined before development iterations began, there was little room for changes in the UI from one iteration to the next. Other tests for the UI included a check that heuristics and navigational design rules were met, and usually after the first iteration, the product was tested with the client.

Project 3

The third project, P3, is based in Australasia and develop software that controls agricultural sorting machines. Their team, P3, consists of five developers, one of whose main interest is user interaction design.

P3 also involved a new version of an existing product. They had adopted a set of XP practices to suit their needs, but the process was still not quite where they envisioned it to be. Up front a written requirements specifications document was created for the basis of the sales agreement with the customer and the user stories and estimates were created from that document. However, during development, UI requirements could still change — sometimes every three months, sometimes daily. The development team gathered just enough information needed to get started on the UI, e.g., what the UI needed to do and what interactions made up a function, and then they evolved the design from that.

The UI design started out as either a pen and paper or PowerPoint prototype, which was then shown to their customers or users for feedback. The prototyping phase sometimes consisted of several iterations of prototyping and obtaining user or customer feedback. Then the prototype was implemented and a concrete piece of software could be shown to the users or customers for feedback. The front end was explained to be well-separated from the back end so that it was easy to switch between different UIs if they changed.

The prototyping phase took place in parallel with the coding phase and were not seen to drive development of the system. The requirements specification document formed the basis of the team's long term plan for the product. This plan was recapped weekly, to determine the team's progress and to decide on the work for the next week. Current bugs, user feedback and other issues with the software would also be discussed on a weekly basis and incorporated into the work for the week. Then the developers went off and did what had been decided on for that week. Interaction design issues were not planned for, it was just seen as part of the work done in an iteration. After the standard XP activities for an iteration had been completed (code, unit test, acceptance test), the UI was run on a simulator and checked. Fixes could result from being run on the simulator (and dealt with as any development fix) or the UI could be left alone for several iterations, without further development or changes.

During development, team members designed UI screens and asked fellow team members for feedback as they went, essentially performing expert reviews. Task-based testing (with users) was more explicitly planned for, but this was performed only when there was enough time. As a result, the feedback from these tests were not coming at the right time for them to be useful or easily incorporated into development, especially when there was a big time lag between the development of the UI and the test. Small fixes resulting from these tests and the expert reviews could be fixed right away, whereas larger fixes were made into new user stories. Sometimes, however, the product was released without the UI being tested at all. The interaction designer saw this as one of the main challenges of incorporating UI design with XP — that the UI activities have trouble keeping up with the fast paced coding iterations. Although usability was seen as beneficial, participants admitted that its techniques were traded off against development time.

The team's lack of experience was cited as a reason why formal usability techniques were not a part of their process. It was one participant's opinion that developers don't need to be the best at solving usability issues, they just need to take into consideration the fact that their products will be used by other people, and that the team's skills and experience would improve over time. The team also had UI guidelines and standards to adhere to during implementation. At the time of the interviews, P3's main aim was to have working software at all times.

Project 4

The final case, P4, involves a software consulting company based in Finland. At the time of the interview, the two participants were not working on the same project, but they described very similar processes in use at their organisation. Both participants on P4 were involved in Scrum development processes.

In P4 the UI team designed the UI 90-95% up front, and their UI design was regarded as the specification for their developers. The UI design process was seen as a separate

process from the development process and Scrum was seen to kick in after UI design had been completed. They preferred to remove the UI design from the implementation process, as the UI design, and feedback about the design, was seen as coming too late in the process. The UI design was used to create the backlog items and estimates for negotiation with the customer. Once the customer had given their approval, implementation began.

Before each iteration, the work for that iteration was planned. Any unfinished features from the previous iteration and the highest priority back-log items became work for that iteration. UI related work during the iteration was 'request-based' — the developer would request help from the UI designer when issues with implementation arose. It was found that some UI decisions could only be made once the internals of the system were known. This meant that some UI design decisions could only be made once implementation had begun. At these times the developers were able to give feedback about UI issues.

There was no user involvement during the implementation process, as all user testing took place up front. The UI designers were approached by the developers during the development process to check usability informally. So the UI was occasionally checked (mainly to ensure that the UI matched the requirements specification) in an unsystematic way. If the customer did not think it was necessary, there was no formal usability testing of the final product. This was seen as a disadvantage, since relying on developers to approach the UI designers with issues, was found to be unreliable. If developers had forgotten to clear some UI issue up with the UI designers, and that issue happened to affect usability, the product was released to the users with that defect. This meant that users suffered with this defect for several months, before the development team got to correct the problem. At the time of the interviews, the UI designer felt that the release plan was very vague. The release dates were fixed, whereas the features that would be included in that release were not. This meant that features were often shipped in a state that the UI designer was not happy with, who would have preferred the release plan to be tied to completed functionality.

5. Results

In this section we present some of the main themes that emerged in our interviews that relate to the issue of up-front UI design. In each of the subsections below, we identify a significant pattern, and provide some relevant passages from the interviews. Passages are identified by the group (P1–4) and the line number in our transcripts.

Iteration Planning Affects UI Design

The first theme that emerged in our interviews relates to iteration planning, determining whether elements of the UI design would be developed in the iteration. At this point, the UI designer may have made some investment in the UI design as lightweight prototypes, but the iteration planning determines what elements will next be developed as functional software. Instead of the planning working on "user stories", therefore, where the UI was involved, the focus was on what we might call "UI stories", where these form coherent elements of a large design.

[P2.43] The designers take a build every morning and test it [the UI]. So we could generate more new cards, but we would negotiate between the developers and the domain expert if whether any new cards would go into the iteration. So, we test every day. We try and get the domain expert to test every few days, or at least once a week so he's checking calculations. And of course we run all our tests as well, which is, I guess, testing as well. We have acceptance tests written by the designers, we have acceptance tests written by the domain expert and then we have the developers' unit tests.

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

[P3.235] They're [user interaction design issues] not explicitly planned for although the usability testing, we try and plan for that. The interaction design is not explicitly planned for; it's just done as part of the iteration.

[P3.289] We'd first make a few design sketches. We quite like to start with either pen and paper or PowerPoint. That can go on at the same time as doing the coding. We want to write really good code, so it's really easy to switch the user interface. It's not too locked to the code. We have the functionality and that's just so we keep different layers. We like to keep the code separate from the user interface so if it changes ideally you can just plug in another one. Then we go on to actually implementing it. It's really hard to describe an iteration. I don't really think of it that way. When I'm working with something I don't think, Ok this is this iteration, this is this iteration. That's why I'm having a hard time describing it, in a sense, as an iteration. Because it becomes just coding because you write something, unit test it and change it, you go back and you acceptance test it, you go back, or you put it on the test simulator on the machine and see how it works and then use the user interface and you might see how it looks and you go back and change it and then leave it for a while. So it's all kind of interleaved.

[P1.177] And then after that [the UI designer doing a walk through of the cards in the iteration planning meeting],

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

One organisation we studied differed in their attitude to the others. It was their goal to complete the UI design before any development began. They then partitioned the design into parts and worked with the developers to schedule implementation of the parts. Despite the apparent difference in attitude, the delivery schedule seems to result in a similar sequence, as we will describe in the following subsections.

[P4.337] Currently what we're trying to come up with is not requirements for a specific iteration. What we're trying to do is actually get the user interface design out of the iteration cycle, so that we would have specified the functionality and how the user interface exactly works and the iterative part would be actually implementing that. So we're kind of using the user interface design as the requirement for the developers.

[P4.338] 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. We have now budget for this user interface design part and that is kind of labeled as requirements analysis. Then when the more typical Scrum tasks come into action are after we've made the interface design and based on that user interface design it's possible to break it into backlog items in Scrum and give estimates of how long this is going to take, and then we make a deal with the customers, "This is the amount that we think this is going to take," and then once they say "Go," we start implementing that. Doing user interface design inside these iterations has proven that it's simply not possible because it comes too late.

[P4.411] No [we didn't use user stories]. We could have used user stories to kind of clarify the things we are doing in each sprint or to use them like place holders for work tasks but we drove down to the actual task level with the planning of each iteration so we kind of skipped that user story phase, because we got this really thick specifications which basically were written down to even technical level. They were too elaborate to be like user stories, they have too much implementation specific design already in them, so we kind of let them be and phased them out and reformulated concrete tasks that would be easy to manage within iterations and easy for the customer to accept and test.

Development Iterations Drive Usability Testing

One of the strong themes that emerged was the relationship between iterations and usability testing. By "usability testing", the UI designers mean testing how easy it is for the end users to work with the software to achieve their goals. This is a well established part of the discipline of human-computer interaction, and there are a variety of techniques that are employed. While some approaches can work with lightweight non-functional prototypes, and some focus on inspection of whether principles and guidelines are followed, the ultimate determination of usability involves actual users working with actual software. The complexity of human behaviour is such that this step cannot successfully be pre-determined, automated, or simulated. The completion of iterations and releases were seen as valuable opportunities to test the usability of the real working software at an earlier point than would have otherwise been possible. Moreover, usability testing was seen as fitting in well with the agile concept of acceptance testing.

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

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

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

[P2.75] We test the build every day, we try and test with our clients up front, so after maybe, the first two week iteration we'll try and test with our client. We do our usability heuristic checking and our domain expert will also test the application. How do we go about the fixes? We mainly card them and negotiate between the developers, ourselves and the domain expert when we put those fixes in, which iteration those fixes will go into.

Again, P4 reflects a different attitude on this matter. It was their position that the UI design process involved iterative development of prototypes, and this process should be

complete before any development began. The usability testing should be completed using lightweight prototypes, and thus not feature in iterative cycles of development. The rationale given was similar to that given by Cooper in the debate organised by Fawcett [3]; Cooper suggests that the implementation process is too slow and expensive to be used in the iterative process of UI design and evaluation. Two issues then arise. One is that the other teams in our study also did iterative UI design and evaluation before development began, but regarded the process as then continuing when the software began development. The other issue is that, as the third quote below illustrates, the UI designers in P4 did become involved in evaluating the results of iterations, but did so informally with the emphasis on checking conformance to their design, rather than considering this usability testing. So, overall, we see the general pattern as having some consistency between all the teams.

[P4.349] Well, the user interface design is kind of a process of its own. It's done iteratively, yes, but the iterations are so short that . . . I can give you pointers to material which show the process we use here . . .

[P4.352] Yes [user testing takes place up front], because it has turned out that that kind of iterating with built code is most expensive and that is also the most stupid thing you can do, because you can do exactly the same kind of testing with users but you can test with lightweight, hand-drawn paper prototypes. That's so much cheaper than building the same system and working code.

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

Usability Testing Results in Changes in Development

The nature of usability testing is that it frequently finds things that need to be changed. In particular, the complex nature of human behaviour means that this is, to some extent, unavoidable, because it is simply not possible to predict how users will behave in new circumstances. So whereas software testing can sometimes identify mistakes in programming, and whereas customers can sometimes change their requirements, usability testing can result in changes even in the absence of mistakes or changes in requirements. This is one reason an iterative process has been universally accepted in UI design. While the iterations in agile development are seen as opportunities for early usability testing, so are following iterations seen as opportunities to change the software to accommodate the results of the testing.

[P2.75] . . . we test the build every day, we try and test with our clients up front, so after maybe, the first two week iteration we'll try and test with our client. We do our usability heuristic checking and our domain expert will also test the application. How do we go about the fixes? We mainly card them and negotiate between the developers, ourselves and the domain expert when we put those fixes in, which iteration those fixes will go into.

[P1.47] Generally if there is a UI issue, it would fall into the category of any other development issue. We want to change x, y and z, so if it does turn out to be expensive like we want a whole new chart that does all these things in 3d and does all these extra fancy bells and whistles and things like that, which is very expensive, then we have to go away and have a huddle about it and see if we can come up with a score, you know, come up with a price. And that will get incorporated into the development iteration in much the same way as any other changes. So we'd have a card following that request that we stick up on the board, and then we'd reshuffle all the other cards.

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

[P3.239] With the user interface you gather as much information as you need to do some kind of real thing and then you put it through an iteration . . . 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. There shouldn't be any up-front design of the underlying code architecture. A lot of what user interface design is about is not so much setting an architecture, well it is a bit, but it's more about finding out about what it has to do and you can not get around the finding out what it has to do aspect and then you can evolve your actual design off of that as you go.

[P3.291] In a way it's [UI issues] scheduled because we have parts of the user interface that we know needs improvement, and a lot of that is based on the feedback that we get, so then we will schedule it in. And sometimes we have one problem on one screen where, for example, the software needs to be restarted if you make this change and the warning was very, very small and so some people wouldn't see it. They'll make this change and then make the mistake of just keep on running the software and go, "Oh, it's not

working." So that was something that we had scheduled to have done for our next official release. It can be scheduled and also parts of the user interface that needs rework.

As we discussed above, P4 took a different approach, and their involvement in the results of development iterations was informal and focused on the issue of conformance to the design specification. In their comments, however, it does appear that this process is not simple, and indeed the developers will approach the UI designers with queries about how the user interaction should work. Moreover, this unofficial approach appears to allow problematic software being deployed. At this point, shown in the third quote below, the participant reflects that more systematic usability testing following iterations might indeed by beneficial.

[P4.384] We also do this unofficially that they [developers] say, "Well, look, come and check this out," so we go and check it during the implementation and try to do this with communication.

[P4.412] The changes [from one iteration to the next] have been pretty manageable. Usually the changes also are driven from the user interface perspective because we start out pretty early with having a kind of PowerPoint sketch and start with that, so part of the changes is working with the HTML prototype where the changes propagate into the development because those two things the HTML prototype is done in parallel to the development and when the development gets to the phase where ... usually we do individual functionalities of features, so the user interface gets developed from the very beginning with the first stories. So there's that kind of change. There is some change also involved of the technical stories but that change isn't usually driven by the customer, it's more driven by the needs of the system or new things learned during the project and risks that didn't get identified in the beginning.

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

Agile Changes the Relationship Between UI Designers and Software Developers

The final theme we identified in our study relates not to any one step in the combined process, but rather to a change in the relationship between UI designers and software developers. Several participants expressed observations about how working together amid the iterations, they realized something new was emerging. This may seem unsurprising, given the basis that both UI design and agile development stress iteration, change, and refinement. But there also seems to emerge a suggestion that these two disciplines may have more to offer each other when they share their iterations.

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

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

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

[P1.111] The other thing, I think, you don't need that many UI designers in XP because the relationship between the UI designer and the developer is just very different. It's not that the UI designer has to come up with the full UI design, everything final, all JavaScript and everything in it. Then goes to the engineers and says ok, implement this and then sees it four months later and has to, like, look through it how does it work? With us it is much more, like, day to day communication. The UI designer actually can get away with not putting all the details and everything into it. Many things just work out during the iteration planning or during development. So I think that actually takes a lot of knowledge away from the UI designer, as he 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 and there are many things that may get dropped.

[P3.234] The idea is to just try and keep the software in a workable state all the time.

6. Conclusions

We have been conducting studies on UI design in agile development, and in this paper we concentrated on the issue of how UI design fits into the structure of agile iterations. Our study was based on interviews with practitioners engaged in agile development of software with significant UI design.

The projects we studied accepted a considerable amount of UI design up front, but then maintained a connection between the UI design process and development iterations. Our study identified several themes that emerged in combining the processes:

- Iteration Planning Affects UI Design
- Development Iterations Drive Usability Testing
- Usability Testing Results in Changes in Development
- Agile Changes the Relationship Between UI Designers and Software Developers

These points show a coherent picture of how the processes can work together for considerable advantage. In particular, even when much UI design has been done up front, interleaving development iterations allows usability testing of working software. Not only does this improve the usability testing, but allows any changes that result to become part of later iterations. Even where a project did not expect to follow this procedure, its advantages were seen. Moreover, the changing relationship suggested between UI designers and software developers shows hope that these practitioners see that working more closely together may assist them in achieving their common goal.

References

[1] S. Chamberlain, H. Sharp, and N. A. M. Maiden. Towards a framework for integrating agile development and user-centred design. In *XP*, pages 143–153, 2006.

[2] J. Earthy, B. S. Jones, and N. Bevan. The improvement of human-centred processes-facing the challenge and reaping the benefit of ISO 13407. *Int. J. Hum.-Comput. Stud.*, 55(4):553–585, 2001.

[3] E. Fawcett. Extreme programming vs. interaction design. *FTP Online*, 2002.

[4] M. Fowler. Is design dead? *Extreme programming examined*, pages 3–17, 2001.

[5] M. Fowler and J. Highsmith. The agile manifesto. *Software Development*, August, 2001.

[6] ISO TC 159/SC 4; ISO Standards ICS: 13.180. *Human-centred design processes for interactive systems*. International Organization for Standardization, Geneva, Switzerland, 1999.

[7] L. Miller. Case study of customer input for a successful product. In *ADC '05: Proceedings of the Agile Development Conference*, pages 225–234, Washington, DC, USA, 2005. IEEE Computer Society.

[8] J. Patton. Hitting the target: adding interaction design to agile software development. In *OOPSLA '02: OOPSLA 2002 Practitioners Reports*, pages 1–ff, New York, NY, USA, 2002. ACM Press.